Metoda wspomagania projektowania zachowania roju dronów z zastosowaniem dwugrafów

Wydział Cybernetyki

Wojskowa Akademia Techniczna

mgr inż. Piotr Cybulski

Promotor:

dr hab. inż. Zbigniew Zieliński, prof. WAT

2 grudnia 2022

Pracę tę dedykuję moim Rodzicom

Spis treści

1 Wprowadzenie **3**

2 Podstawowe pojęcia i definicje **17**

2.1 Zbiory i funkcje 17

2.2 Grafy 22

2.3 Dwugrafy 24

2.4 Dwugrafowe systemy reakcyjne 40

2.5 Systemy przejść 45

3 Metodyka projektowania zachowania roju dronów z użyciem dwugrafów **49**

3.1 Przyjęte założenia względem projektowanych systemów 49

3.2 Przestrzeń stanów 52

3.3 Polityka zachowania 56

3.4 Harmonogram działań 60

3.5 Całościowy obraz metodyki 64

4 Metoda weryfikacji modeli systemów wieloagentowych **70**

4.1 Warunki poprawności modelu MAS 70

4.2 Algorytm weryfikacji modelu 72

4.2.1 Etap 4 74

4.2.2 Etap 3 78

4.2.3 Etap 2 85

4.2.4 Etap 1 91

5 Zastosowanie metodyki projektowania zachowania roju dronów **95**

5.1 Działania kooperacyjne w trakcie misji 95
5.2 Weryfikacja modelu 101
5.3 Wykrycie błędów w modelu 112
5.4 Skalowanie projektu 117
5.4.1 Skalowanie w poziomie 118
5.4.2 Skalowanie w pionie 121
5.5 Weryfikacja wymagań pozafunkcjonalnych 130
5.6 Praktyczne ograniczenia w wykorzystaniu opracowanej metodyki . 134

6 Podsumowanie i kierunki dalszych prac 144

Wykaz skrótów 148

Słownik terminów 150

Bibliografia 153

Spis rysunków 163

Spis tabel 166

Załącznik A Weryfikacja wyników eksperymentów 169

Załącznik B Sposób przekształcenia TBRS w TTS 181

Załącznik C Sposób przekształcenia TTS w SS 187

Załącznik D Konstrukcja stanu scenariusza 192

Załącznik E Analiza złożoności wybranych algorytmów 197

Załącznik F Rozszerzanie marszruty 203

Rozdział 1

Wprowadzenie

Wykorzystanie narzędzi przez człowieka do łatwiejszego wykonania prac trudnych lub niebezpiecznych nie jest niczym nowym. Wraz z rozwojem ludzkości i zwiększeniem skomplikowania zadań jakie ludzie codziennie wykonują pojawił się pomysł opracowania narzędzi, które będą w stanie działać do pewnego stopnia samodzielnie, a przez to muszą być zdolne do percepcji otoczenia oraz samodzielnej z nim interakcji.

Byty posiadające zdolność percepcji otoczenia oraz wchodzenia z nim w interakcję są określane w informatyce jako *agenci* [72]. Mogą to być zarówno ludzie, jak i roboty, ale również programy komputerowe [83]. Zbiór tak zdefiniowanych agentów działających w jakimś celu we współdzielonym otoczeniu jest określany jako *system wieloagentowy* (*ang. multi-agent system*), w skrócie MAS.

Systemy wieloagentowe, w których agenci są robotami to *systemy wielorobotowe* [67, 70] (*ang. multi-robot system* - MRS). Rozpatrywanie tego wariantu MAS jest trudniejsze niż pozostałych [46] z kilku powodów. Przede wszystkim sposób komunikacji pomiędzy agentami, postrzeganie i oddziaływanie na otoczenie są znacznie trudniejsze do uwzględnienia w modelu. Czynności te wymagają wykorzystania sensorów oraz efektorów, które mają swoje parametry użytkowania oraz ograniczenia. Mnogość możliwości w projektowaniu MRS oraz trudności z nimi związane spowodowały, że powstało wiele sposobów klasyfikacji tego typu projektów [34, 32, 46]. Do typowych kryteriów, w oparciu o które klasyfikowane są projekty MRS, można zaliczyć: sposób komunikacji, stopień koordynacji działań robotów oraz sposób sterowania robotami. W literaturze wydaje się panować

konsensus, że typowe metody projektowania MAS są niewystarczające, gdy zaangażowane są w nie roboty. Jako przykład sposobu projektowania MRS, który nie jest często spotykany w przypadku innych typów MAS można wskazać systemy, w których współpracujący agenci są nieświadomi siebie nawzajem[34] (*ang. cooperative unaware*).

Roje robotów (RS) to potoczna nazwa wariantu systemu wielorobotowego, którego sposób działania przejawia znamiona inteligencji rojowej (*ang. swarm intelligence*). Nie ma jednak konsensusu, jakie cechy musi posiadać MRS, aby można go było bezsprzecznie uznać za rój robotów [10]. Niektóre prace [70] używają zamiennie pojęć rój robotów i system wielorobotowy. Powodem tego może być fakt, że rozpatrywanie agentów, których zachowanie odzwierciedla to obserwowane w naturalnych stadach, czy rojach nie jest koncepcją nową i sięga aż do „zwykłych" systemów wieloagentowych [59]. Wymieniane w literaturze [42, 73, 63] cechy charakterystyczne roju można podzielić na te, które są przypisywane jego członkom, jak i całemu rojowi. Do tych pierwszych można zaliczyć lokalne (jako przeciwieństwo globalnych) zdolności komunikacji i percepcji oraz autonomiczność. Czasami zaliczany jest tutaj również brak celu *a priori* [52] dla agentów. Główne trzy cechy roju robotów wymieniane w literaturze to: skalowalność (*ang. scalability*), odporność na awarie (*ang. robustness*) oraz adaptacyjność do różnych zadań oraz warunków w jakich są wykonywane (*ang. flexibility*). Również często wymienianymi cechami charakterystycznymi całego roju jest brak globalnego źródła wiedzy (tj. jednakowego i dostępnego dla wszystkich) oraz duża liczba członków wchodzących w jego skład. Sugerowana jest również homogeniczność, a przynajmniej występowanie homogenicznych podgrup w jego ramach. Właściwością wymienianą przez wszystkie prace dotyczące rojów robotów jest to, że zachowanie całego roju „wyłania się" z lokalnych działań jego członków. Należy tutaj podkreślić różnicę pomiędzy właściwościami, a parametrami. Przez właściwość rozumieć będziemy coś, co jest cechą, która nie jest sterowalna. Jako przykład przez analogię można użyć piłkarza, którego parametrami jest to jak szybko biega i kopie piłkę, a właściwością jest jego ocena jako zawodnika. Przez lokalne działania będziemy rozumieć czynności, które mogą być wykonywane przez członków roju niezależnie od postawionego zadania, które pojedynczo mogą nie przybliżać

w oczywisty sposób do realizacji postawionego zadania. Zachowanie określane jako „wyłaniające się" (*ang. emergent behavior*), czasami zwane również kolektywnym, jest definiowane w literaturze jako bardziej złożone niż sposób działania dowolnego pojedynczego elementu roju.

W ramach tej pracy *rój robotów* będzie rozumiany jako system wielorobotowy o następujących cechach:

1. Możliwości działania członków roju są definiowane niezależnie od postawionego zadania oraz rozmiaru samego roju.

2. Rozpatrywany MRS jest zdolny do wyłonienia zachowania kolektywnego. W praktyce oznacza to, że pożądane zachowanie całego roju powinno powstać z ciągu działań jego elementów.

Pierwsza z cech gwarantuje, że rój jest zarówno skalowalny (w rozumieniu liczby członków), jak i może zostać przystosowany do realizacji różnych zadań. Druga cecha jest traktowana jedynie jako założenie, ponieważ pożądane zachowanie jest przedmiotem projektowania. Gdyby istniała pewność, że pożądany sposób działania roju istnieje, to wystarczyło by wykorzystać to, które służy jako dowód. W tej pracy awaryjność członków roju nie jest brana pod uwagę. Jest to świadome ograniczenie poruszanego tutaj zagadnienia, ponieważ przykłady dostępne w literaturze [84, 12] wskazują, że założenie odporności rojów robotów na awarie nie ma przełożenia na rzeczywistość[1].

Roje robotów, których agentami są bezzałogowe pojazdy latające (BPL) będą w dalszej części pracy określane *rojami dronów*, *rojami UAV* lub *rojami BPL*. Wszystkie te określenia będą traktowane jako równoważne.

Do głównych cech dronów wyróżniających je na tle innych typów robotów, które również mogą stanowić część roju należy zaliczyć:

- wysoką mobilność;

- duże możliwości komunikacji między sobą, a także z infrastrukturą naziemną;

[1]Szczególnie interesujące w tym względzie są wyniki z [84], które sugerują, że częściowe awarie członków roju (np. w postaci awarii tylko uniemożliwiającej przemieszczenie) są bardziej szkodliwe dla roju niż zniszczenie tychże członków.

- krótki czas działania;
- ograniczone możliwości obliczeniowe.

Pierwsze dwie cechy wynikają z możliwości poruszania się dronów w trzech płaszczyznach (w odróżnieniu od dwuwymiarowych zdolności przemieszczania się np. łazików). Dzięki temu, są one w stanie dostać się do miejsc nieosiągalnych dla znacznej części innych typów robotów. Również dzięki możliwości przemieszczania się w trzech wymiarach drony mogą komunikować się, omijając znaczną część przeszkód terenowych. Jedną z powszechnie znanych cech dronów jest ich krótki czas działania. Wynika ona zarówno z aktualnych możliwości przechowywania energii, jak i z samego faktu, że drony przemieszczają się drogą powietrzną. Z ograniczeń technologicznych obecnie wykorzystywanych akumulatorów, a także z konieczności wykorzystania (znacznej) części zgromadzonej w nich energii na przemieszczenie drogą powietrzną wynikają ograniczone możliwości obliczeniowe pojedynczego drona.

Pojedyncze roboty zdolne do postrzegania i wchodzenia w interakcję z otoczeniem, czyli agenci, znajdują aktualnie zastosowanie w wielu zadaniach. Należy tu przynajmniej wymienić takie prace jak eksploracja głębin oceanicznych [57], wsparcie przy ocenie stanu floty wodnej [33], unieszkodliwianie materiałów wybuchowych [40], czy monitoring i badanie flory zarówno trudno dostępnej jak i tej, która wymaga częstego próbkowania [82, 65].

Systemy wielorobotowe pomimo trudniejszych w projektowaniu, znajdują zastosowanie wszędzie tam, gdzie mogą efektywniej wykonać zadania, aktualnie realizowane przez pojedynczego robota. Przede wszystkim mówimy tutaj o zastąpieniu schematu *jeden operator-jeden robot* schematem *jeden operator-wiele robotów*. Ten typ zastosowań MRS jest powszechnie sugerowany w zadaniach, w których udział człowieka stanowi dla niego zagrożenie lub polega na wykonaniu prostych czynności, ale na dużym obszarze [11, 41] - RISE (*ang. Risky Interventions and Surveillance of the Environment*). Kolejnym typem zadań, do których są angażowane systemy wielorobotowe jest przemieszczenie przedmiotów [86, 38]. Można tu wymienić transportowanie zarówno w formie przesuwania, jak i zaawansowanych form przemieszczenia, uwzględniających podnoszenie obiektu. Opracowywane są również sposoby wykorzystania systemów wielorobotowych w

zadaniach polegających na szukaniu oraz obserwowaniu obiektów [3, 51], zarówno poruszających się szybciej jak i wolniej niż roboty. Rozpatrywane są także zastosowania MRS w takich zadaniach jak gry zespołowe [55, 47], czy wsparcie osób niepełnosprawnych [8, 88]. Roje robotów będące systemem wielorobotowym o pewnych cechach i względem których przyjmowane są założenia, że posiadają określone właściwości znajdują zastosowanie w wariantach zadań realizowanych przez MRS.

W zdecydowanej większości publikacji [66, 63, 42, 15] dotyczących rojów robotów, zadania, które mogą być wykonywane przez rój przypisywane są do jednej z trzech kategorii: organizacji w przestrzeni, zbiorowego przemieszczenia lub kolektywnego podejmowania decyzji.

Zadania pierwszej grupy polegają na uzyskaniu przez rój jakiejś cechy przestrzennej. Przykładem może być żądana odległość pomiędzy robotami. Do zadań w tej kategorii należą przede wszystkim zbiórka w punkcie (agregacja), rozproszenie, pokrycie obszaru i utworzenie formacji. Charakterystyczne dla tych zadań jest to, że celem jest zmiana samego roju.

Do grupy zbiorowego przemieszczenia zaliczają się wszystkie zadania polegające na ominięciu przeszkody lub znalezieniu rozrzuconych obiektów. Najczęściej rozpatrywanymi w literaturze zadaniami tej grupy są: eksploracja, poszukiwanie (*ang. foraging*) polegające na gromadzeniu znajdujących się na mapie obiektów, grupowa nawigacja polegająca na utworzeniu i zmodyfikowaniu w razie potrzeby szyku zmierzającego w wybranym przez rój kierunku, grupowy transport polegający na przemieszczeniu obiektów, który wymaga w tym celu zaangażowania co najmniej kilku robotów.

Ostatnia grupa składa się z zadań polegających na podjęciu decyzji przez członków roju dotyczącej całego roju. Roboty podejmują decyzję w oparciu o własną percepcję oraz informacje uzyskane od innych robotów. Tego typu zadania pojawiają się wszędzie tam, gdzie nie ma dostępu do globalnej współdzielonej informacji. Do zadań tej kategorii należą m.in.: ustalenie konsensusu polegające na podjęciu jednakowej decyzji przez cały rój, przydział zadań polegający na przypisaniu każdemu robotowi w roju pewnego zdania z listy dostępnych, ustalenie lokalizacji (zarówno poszczególnych członków jak i całego roju w przestrzeni).

Inne sposoby podziału zadań stawianych przed rojem występujące w literaturze [73] kategoryzują zadania na takie, które dotyczą: właściwości przestrzennych otoczenia, w którym działa rój; które wymagają uwzględnienia strat w roju w trakcie wykonywania zadania; mogące angażować różnych rozmiarów roje wraz z postępem wykonywania zadania; które wymagają nadmiarowości (redundancji) jako formy zabezpieczenia przed awarią systemu, który tworzy rój.

W literaturze dotyczącej rojów BPL [71, 44, 89, 78, 85] jest wyraźny podział na zastosowanie ich w sferze cywilnej i wojskowej. Do zadań w sferze cywilnej, które są stawiane przed rojami robotów najczęściej zaliczane są: inspekcja infrastruktury, wsparcie na budowie, wszystkie zadania typu SAR, transport towarów, monitorowanie wskazanego obszaru, wsparcie rolnictwa, przeciwdziałanie i walka z katastrofami naturalnymi. Do zastosowań cywilnych rojów BPL, które mają na celu wykorzystać cechy dronów wyróżniające je od innych typów robotów, należy przede wszystkich zaliczyć zaangażowanie ich w ramach inteligentnych miast (ang. *smart cities)* oraz przy ustanawianiu sieci ad hoc nad obszarami nieposiadającymi przeznaczonej do tego infrastruktury. Drony wchodzące w skład roju wykorzystywanego w ramach inteligentnego miasta pełniłyby różne role, w zależności od aktualnej sytuacji i potrzeb z niej wynikających. Przykładowo, w [61] wymieniono kilka ról, które mogłyby być pełnione przez członków roju BPL. Wskazano tam między innymi monitorowanie i raportowanie incydentów na drodze, w tym zastąpienie służb policyjnych w części działań związanych z kontrolą drogową, a także bycie częścią i wsparcie infrastruktury niezbędnej do funkcjonowania inteligentnego miasta. Roje dronów mogą zostać również wykorzystane do konstrukcji latających sieci ad hoc (ang. *flying ad hoc networks* - FANET)[64], która nie wymaga infrastruktury naziemnej na obszarze, na którym konstruowana sieć ma działać. Jest to wariant przenośnych sieci ad hoc (ang. *mobile ad hoc networks* - MANET), który wykorzystuje cechy charakterystyczne dronów (w szczególności ich mobilność). W odróżnieniu od sieci typu VANET (ang. *vehicle ad hoc networks*) mają one umożliwić komunikację pomiędzy obszarami, na których występują przeszkody terenowe oraz wymagane są częste zmiany topologii sieci[7]. Powyższe zadania nie zostały pogrupowane zgodnie z kategoriami zadań MRS, ponieważ przeważająca większość z nich wymaga rozwiązania pro-

blemów z więcej niż jednej grupy. Publikacje dotyczące zastosowania rojów UAV w sferze wojskowej, czy szeroko pojętego bezpieczeństwa, skupiają się na przechwytywaniu szkodliwych dronów w przestrzeni publicznej przy użyciu roju [16], zakłócaniu działających rojów o wrogich zamiarach [45]. Należy podkreślić, że publikacji poświęconych zastosowaniom rojów dronów w obszarze bezpieczeństwa jest znacznie mniej niż w sferze cywilnej. Poza kwestią jawności wyników badań powodem mogą być wątpliwości przewagi rojów BPL nad pojedynczym zaawansowanym dronem bojowym [85], przy jednoczesnym względnie łatwym sposobem walki z nimi [45, 87].

Przedmiotem tej pracy jest inżynieria roju robotów. Korzystając z definicji przedstawionej w [15][2] celem inżynierii rojów robotów jest opracowanie sterowalnego roju działającego w przewidywalny sposób, z jasno zdefiniowanymi celami i ograniczeniami. Zadanie to można zdekomponować na dwa problemy. Pierwszym jest opracowanie systemu wielorobotowego, który ma pełnić rolę docelowego roju. Drugim jest opracowanie sposobu działania wspomnianego MRS. Celem tej pracy jest przedstawienie sposobu opracowania zachowania dla istniejącego systemu wielorobotowego, który posiada wcześniej wymienione cechy roju. Przez zachowanie roju robotów będziemy w tej pracy rozumieć sposób jego funkcjonowania, który wyłania się z działania jego członków opartego na elementarnych czynnościach. Właśnie ten indywidualny sposób działania członków roju jest przedmiotem projektowania, a zachowanie całego roju będące synergią działań składowych jest jego celem. Mając do czynienia z więcej niż jednym robotem, zgodnie z terminologią zaproponowaną w [19] będziemy określali zachowanie całego roju zachowaniem zbiorowym. Ponadto, jako że rój zawsze będzie miał realizować jakiś cel, będziemy nazywać działania składowe elementów roju współpracą.

Zarówno we wczesnej literaturze dotyczącej MRS/RS [19, 15], jak i aktualnych przeglądach stanu wiedzy [52, 42], znaczna część wymienianych trudności związanych z projektowaniem zachowania roju robotów pokrywa się. Najczęściej

[2]Autorzy wspomnianej publikacji powołują się na definicję z rozprawy doktorskiej [49]. Rozprawa ta w wersji online dostępnej pod adresem: `http://thesis.library.caltech.edu/6105/` nie zawiera wspomnianego cytatu. Definicja ta, jak i sam artykuł jest jedną z najczęściej cytowanych w literaturze dotyczącej rojów robotów, a ponadto autor tej rozprawy się z nią zgadza.

wymieniane trudności związane są z doborem, modelowaniem i sposobem wykorzystania parametrów robotów wchodzących w skład roju. Niektóre możliwości ułatwiają zaprojektowanie zachowania dla całego roju, a niektóre znacząco to utrudniają, ale są niezbędne do wykonania postawionego zadania. W tej rozprawie skupiono się na problemach teoretycznych dotyczących projektowania zachowania całego roju robotów, abstrahując od konkretnej technologii, w oparciu na której członkowie roju będą działali. Problemem pomijanym w ramach tego założenia będzie np. koordynacja ramion robota mającego podnieść przedmiot albo regulacja prędkości obrotów poszczególnych śmigieł BPL, niezbędna do jego przemieszczenia w oczekiwanym kierunku. Z kolei przykładem możliwości, której posiadanie przez członków roju możemy rozpatrywać będzie to, czy i w jakim stopniu uwzględniają pozostałe elementy roju. Parametry członków roju można podzielić na trzy grupy.

Pierwszą grupę stanowią cechy, które definiują w jaki sposób funkcjonuje pojedynczy członek roju. Uwzględniamy tutaj zarówno to, czy jest on świadomy istnienia pozostałych robotów, jak i to, w jaki sposób wchodzi w interakcję z otoczeniem (celowo czy przypadkowo).

Drugą grupę stanowią cechy służące do zdefiniowania sposobu wchodzenia w interakcję między członkami roju. Przykładem takich cech mogą być założenia względem zasięgu oraz topologii komunikacji. W [32] wyróżniono trzy wartości na osi zasięgu komunikacji: brak komunikacji, komunikacja na krótką odległość oraz komunikacja bez ograniczeń dystansowych. Z kolei w tej samej publikacji rozróżniane topologie komunikacji to: komunikacja peer-to-peer (1-1), gdzie każdy robot może nawiązać bezpośredni kontakt z dowolnym innym (dostępnym po uwzględnieniu założeń dotyczących zasięgu komunikacji); hierarchiczna, gdzie w ramach roju część jego członków pełni rolę przekaźników (można ich traktować jak routery w roju); grafowa, gdzie każdy robot może komunikować się z podzbiorem reszty członków roju zdefiniowanym jako graf.

Ostatnia grupa cech skupia się na możliwościach komunikacji roju „ze światem zewnętrznym". W [52] wskazano dwa główne powody, dla których taka komunikacja może być pożądana: umożliwienie analizy sposobu działania roju obserwatorowi nie mającemu kontaktu wzrokowego z rojem; umożliwienie operatorowi

sterowanie rojem w trakcie jego działania. Ponadto powszechnie wskazywaną trudnością w projektowaniu zachowania rojów robotów, a także MRS oraz MAS, jest wybór metody „uczenia" jego członków, w jaki sposób mają działać. W [15] uznano następujące cztery obszary projektowania RS jako wymagające rozwinięcia, aby roje robotów mogły znaleźć zastosowanie poza środowiskiem laboratoryjnym:

- Sposoby zbierania oraz formalnego definiowania wymagań. Autorzy nie określili, czy metody analizy wymagań, które są stosowane w innych obszarach znajdują również zastosowanie w obszarze RS.
- Weryfikacja i porównanie opracowanych rojów. We wspomnianej publikacji wskazano, że nie istnieją uznawane powszechnie miary, w oparciu o które można porównać roje opracowane z użyciem różnych metod. Ponadto podkreślono, że tylko dla dwóch typów zadań (zbieranie przedmiotów oraz konstruowanie nowych) są opracowane testy, mające zweryfikować opracowany rój przed wykonaniem docelowego zadania.
- Nadzorowanie i interakcja z rojem w trakcie wykonywania zadania.
- Metodyki projektowania zachowania roju niezależne od stawianych zadań.

W celu przezwyciężenia wymienionych wyżej problemów opracowano wiele metod i metodyk projektowania zachowania roju robotów. W tej pracy przez metodę należy rozumieć algorytm, czyli uporządkowany zbiór kroków służących osiągnięciu jakiegoś celu. Z kolei metodyka będzie rozumiana jako metoda wraz ze sposobem, jak przekształcić elementy świata rzeczywistego w struktury, na których operuje metoda.

Najczęściej stosowany podział istniejących metod projektowania rozróżnia metody „od szczegółu do ogółu" (*ang. bottom-up*) oraz „od ogółu do szczegółu" (*ang. top-down*). Pierwsza grupa z definicji [21] rozpoczyna od ustalenia możliwości elementów projektowanego systemu (w naszym przypadku będą to drony), a następnie opracowywany jest sposób ich działania, jako zbioru interakcji pomiędzy komponentami oraz komponentami i otoczeniem. Początkowe możliwości robotów w tym sposobie projektowania nie uwzględniają oceny otoczenia. Alternatywnym sposobem projektowania jest rozpoczęcie od określenia wymagań

docelowego stanu (np. w postaci funkcji oceny stanu systemu), a następnie dokonywana jest próba rozproszenia globalnej oceny na pojedyncze roboty. Każdy robot może oceniać fragment otoczenia, który zakładamy, że jest w stanie obserwować. Indywidualne oceny fragmentów otoczenia mają umożliwić działanie robotów w taki sposób, aby wartość funkcji oceny stanu systemu była satysfakcjonująca. Metody „od szczegółu do ogółu" są dominujące w literaturze. W [15] dokonano podziału metod projektowania zachowania na takie, w których zarówno algorytm, jak i wartości jego parametrów są jawnie dobierane przez projektanta oraz na takie, gdzie algorytm jest „dostrajany" przy użyciu metod uczenia maszynowego. Nawiązując do wcześniej zaproponowanego sposobu klasyfikacji, te pierwsze są w znacznej większości, według autorów wspomnianej publikacji, metodami „od szczegółu do ogółu". Z kolei zaangażowanie uczenia maszynowego zazwyczaj jest utożsamiane przez nich z tym, że metoda należy do grupy top-down. W tej samej publikacji zaproponowano podział istniejących metod modelowania roju na dwie grupy. Pierwsza grupa, w której wszyscy członkowie roju i interakcje między nimi i otoczeniem są uwzględniane w modelu określa modele do niej zaliczane jako „mikroskopowe". Druga grupa określająca modele „makroskopowymi" skupia te z nich, w których rój jest traktowany jako całość, bez rozróżnienia jego elementów składowych. W literaturze [66, 42, 52] dominuje jednak klasyfikacja istniejących metod projektowania RS oparta na obszarach wiedzy, z których narzędzia formalne zostały wykorzystane do modelowania elementów roju oraz interakcji między nimi. Do najczęściej wymienianych źródeł inspiracji, z których narzędzia formalne są w tym celu wykorzystywane można zaliczyć:

- Teoria sterowania w połączeniu z teorią grafów i sieci - modele inspirowane tymi obszarami wiedzy zazwyczaj przedstawiają elementy roju jako systemy dynamiczne (w definicji cybernetycznej z [39]), czyli takie, w których istotną rolę odgrywają zarówno zdarzenia wcześniejsze, jak i opóźnienia czasowe działań podejmowanych przez system. Zdolności komunikacyjne oraz percepcji otoczenia są przedstawiane w takich modelach z użyciem pojęć teorii grafów. Kompleksowe wprowadzenie do modelowania MRS w ten sposób przedstawiono w [17]. Można spotkać opinie [15, 52], że ze względu na mnogość przyjmowanych założeń, które bardzo prawdopodobne, że zo-

staną naruszone w implementacji fizycznej, ten sposób nie nadaje się do praktycznego modelowania roju dronów. Ten problem jest określany w literaturze jako różnica pomiędzy wynikiem teoretycznym, a praktycznym (ang. *reality gap*) [35].

- Fizyka - modele, w których do reprezentacji sposobu działania członków roju używane są analogie do zjawisk fizycznych określane są w literaturze modelami inspirowanymi fizyką. W przeważającej większości członkowie roju są w takich modelach porównywani do cząsteczek, pomiędzy którymi oddziaływania determinują sposób funkcjonowania roju. Jednym z częściej przytaczanych przykładów tego typu modeli jest [81]. Autorzy przedstawili w nim metodę projektowania zachowania roju w sposób umożliwiający realizację zadań z grup organizacji przestrzennej oraz zbiorowego przemieszczenia. Cechą charakterystyczną tego typu modeli jest to, że skupiają się na pasywnych oddziaływaniach pomiędzy członkami roju. Tego typu zachowanie roju można zaklasyfikować do grupy słabo skoordynowanych, albo nieskoordynowanych według definicji z [46]. Jak sami autorzy podkreślają daje to dużą odporność na awarie, ale kosztem złożoności zadań postawionych przed rojem (wymieniają choćby niezdolność wykonania zadań wymagających planowania).

- Biologia - ostatnim, a jednocześnie najpopularniejszym sposobem modelowania rojów robotów jest próba odwzorowania zachowań zwierząt przy użyciu aparatu matematycznego. Chyba najczęściej przytaczaną pracą z tej kategorii jest [20]. Podobnie jak w przypadku modeli inspirowanych zjawiskami fizycznymi, tutaj również wyniki umożliwiają realizację zadań z grupy zorganizowanego przemieszczenia oraz organizacji przestrzennej. Kluczową różnicą pomiędzy tym sposobem modelowania, a tymi inspirowanymi fizyką jest sposób komunikacji pomiędzy członkami roju. W przypadku modeli inspirowanych szeroko rozumianymi zjawiskami biologicznymi komunikacja pomiędzy elementami roju ma charakter aktywny. Należy przez to rozumieć, że inaczej niż ma to miejsce w modelach inspirowanych fizyką za każdą wymianą komunikatów stoi intencja, któregoś z porozumiewających się. Jest to trudniejsze w realizacji, jednak daje większą adaptacyjność,

co można zauważyć choćby we wspominanej pracy, gdzie zmiana wartości parametrów roju pozwala na całkowitą zmianę zachowania roju przy jednoczesnym zachowaniu przewidywalności, tego w jaki sposób rój będzie działał. Modele tego typu pozwalają opracować zachowanie roju, które będzie kwalifikowało się do miana silnie skoordynowanego według definicji z [46].

Rzadziej przytaczanym obszarem, z którego wykorzystywane są pojęcia i narzędzia formalne do modelowania roju są bezkształtne systemy komputerowe (*ang. amorhphous computing*) [1]. Są to systemy, w których zbiór zawodnych, przemieszczających się, zdolnych do komunikacji elementów ma działać w pożądany sposób. O ile nie można powiedzieć, że podejście to zostało porzucone [6], to zdecydowanie jest to obszar, który zasługuje na uwagę ze względu na koncepcyjne podobieństwo do tego, co jest utożsamiane z rojem.

Ostatnią grupą sposobów projektowania zachowania rojów robotów są metodyki [36, 14, 43, 13, 56], w których nie można wskazać jednego dominującego obszaru nauki, z którego zaczerpnięte są narzędzia formalne. Ponadto metodyki projektowania zachowania roju robotów starają się zautomatyzować ten proces i podkreślają potrzebę jego standaryzacji, w celu umożliwienia weryfikacji rozwiązań. Do oceny poziomu automatyzacji procesu projektowania używana w tej pracy będzie dziesięciostopniowa skala zaproponowana w [79]. Proces na pierwszym poziomie automatyzacji to taki, w którym człowiek wykonuje całą pracę intelektualną, a system komputerowy ma tylko wykonać zadane instrukcje. W procesie na najwyższym (dziesiątym) poziomie automatyzacji system komputerowy podejmuje decyzje, czy oraz jak wykonać postawione zadanie. To, czy człowiek (operator) zostanie poinformowany o rezultatach również jest w gestii komputera. W tej pracy skupiono się na poziomach 7 oraz 8, czyli takich gdzie polecenie (opracowania sposobu zachowania roju) musi zostać wykonane przez system komputerowy, a informacje o wyniku będą albo zawsze zwracane (poziom 7), albo tylko na żądanie operatora (poziom 8). Obecnie najlepsze rezultaty w zakresie standaryzacji projektowania działania roju robotów przedstawia metodyka omówiona w [36]. Na jej podstawie powstało kilka rozszerzeń, których celem jest zarówno poprawa jakości opracowanego sposobu zachowania roju [43, 56], jak i

ujednolicenie sposobu projektowania różnych typów zadań [13].

Nadal istnieje szereg trudności związanych z projektowaniem rojów robotów czy rojów dronów. Do trudności tych należy zaliczyć przede wszystkim modyfikację roju lub zadania w trakcie projektowania. Bardzo istotny jest również sposób weryfikacji wymagań oraz etap projektu, na którym ten proces jest wykonywany. Metody formalne wydają się jedynym sposobem rozwiązania powyższych trudności, jednak należy pamiętać, że sformalizowanie procesu projektowania nie jest celem, a środkiem. Podejście formalne powinno umożliwić wczesną weryfikację przynajmniej wykonalności postawionego przed rojem zadania. Z kolei, na dalszych etapach projektowania metody te powinny pozwolić na coraz bardziej szczegółową weryfikację przed-wdrożeniową opracowanego zachowania. Bardzo pożądaną cechą ze względu na specyfikę rojów robotów jest również możliwość nieinwazyjnego w projekt skalowania rozmiaru rozpatrywanego zadania, a także samego roju.

Dwugrafy (*ang. bigraphs*) są formalizmem wprowadzonym przez Robina Milnera [62] umożliwiającym modelowanie systemów, w których istotną rolę odgrywa zarówno rozmieszczenie elementów, jak i połączenia między nimi. Tym, co wyróżnia język dwugrafów jest to, że wszystkie struktury skonstruowane z jego użyciem mogą być przedstawione w formie graficznej oraz to, że umożliwia on w łatwy sposób rozbudowę już zamodelowanych systemów. Możliwe jest również zdefiniowanie dynamiki projektowanego systemu przy użyciu struktur określanych jako reguły reakcji. Formalizm ten został w swojej finalnej postaci opublikowany w 2008 roku, a już doczekał się kilku rozszerzeń [4, 54, 77]. Dwugrafy udowodniły swoją przydatność w projektowaniu systemów internetu rzeczy [80, 5] oraz sieci komputerowych [18]. Jako narzędzie formalne wspomagające wykonanie prac związanych z projektowaniem zachowania roju robotów dwugrafy zostały dotychczas tylko pobieżnie zbadane. Istniejące prace ograniczają się do sposobów projektowania systemów wieloagentowych z wykorzystaniem dwugrafów [58, 31, 30]. Nie przedstawiają one zazwyczaj formalnego sposobu zdefiniowania zachowania dla elementów systemu, w oparciu o skonstruowany model. Istniejące prace [68], które uwzględniają i ten aspekt projektowania MAS robią to wykorzystując dodatkową warstwę abstrakcji (zazwyczaj są to aktorzy [2]). Ponadto

są obecnie równolegle rozwijane narzędzia [37, 76], które opierają się na tym formalizmie. Pozwala to mieć nadzieję, że dwugrafy będą w przyszłości liczącym się narzędziem formalnym w projektowaniu i weryfikacji różnego typu systemów.

W oparciu o przytoczoną literaturę oraz aktualny stan prac nad wykorzystaniem rojów robotów do celów zarówno cywilnych, jak i wojskowych postawiono następującą tezę:

> *Dwugrafy pozwalają na konstrukcję wystarczająco ekspresyjnych modeli, aby na ich podstawie skutecznie projektować zachowanie dla elementów roju dronów.*

W celu weryfikacji postawionej tezy wskazano następujące cele, których wyniki opisano w niniejszej pracy:

1. Opracowanie metodyki projektowania zachowania elementów RS, która opiera się na języku dwugrafów. Wdrożenie rezultatu zastosowania metodyki powinno gwarantować wykonanie zadania w bezpieczny sposób.

2. Opracowanie metody weryfikacji wyników opracowanej metodyki.

3. Opracowanie oprogramowania, które pozwoli zautomatyzować zarówno proces projektowania zachowania, jak i proces weryfikacji uzyskanych za jego pomocą wyników.

Rozdział 2

Podstawowe pojęcia i definicje

W tym rozdziale zostaną wprowadzone pojęcia niezbędne do zrozumienia dalszej części pracy, a których definicje mogą się różnić w zależności od materiałów źródłowych. Jednocześnie terminologia tutaj wprowadzona nie jest wynikiem prac związanych z dysertacją, a co najwyżej rozszerzeniem już istniejącego aparatu formalnego.

2.1 Zbiory i funkcje

Zbiór jest jednym z podstawowych pojęć w matematyce [69] i od niego zostaną rozpoczęte definicje formalne. Przedmioty, które należą do danego zbioru określane są jego *elementami*. Jeżeli element a należy do zbioru $Y = \{a, b, c\}$ to będzie to zapisywane jako $a \in Y$. Zbiór pusty, czyli nie zawierający żadnego elementu oznaczany będzie symbolem $\emptyset$. Zbiory będą w tej pracy definiowane na dwa sposoby. Pierwszy sposób określa zbiór w odniesieniu do już wcześniej zdefiniowanych np. $X = \mathbb{R}$ oznacza, że zbiór X jest równy zbiorowi liczb rzeczywistych. Drugi sposób opiera się na funkcjach zdaniowych i np. $X = \{x \mid x \in \mathbb{R} \wedge x + 1 > 0\}$ definiuje zbiór, którego elementami są liczby rzeczywiste większe od -1. Do operacji na zbiorach, które będą używane w dalszej części pracy należą:

- Część wspólna zbiorów A i B, oznaczana przez $A \cap B$ jest zbiorem elementów, które należą jednocześnie do zbioru A i B.
  Przykładowo $\{x, y\} \cap \{y, z\} = \{y\}$.

- Operacja sumy zbiorów, które mogą zawierać wspólne elementy będzie ozna-

czana symbolem $\cup$. Przykładowo sumą zbiorów $A = \{x, y\}$ i $B = \{y, z\}$ jest zbiór $C = \{x, y, z\}$.

W przypadku, gdy składowymi sumy będą n-tki uporządkowane składające się z takiej samej liczby zbiorów operacja $(S_1, S_2, \ldots, S_n) \cup (S'_1, S'_2, \ldots, S'_n)$ daje w wyniku n-tkę uporządkowaną, której elementami są zsumowane zbiory tj. $(S_1 \cup S'_1, S_2 \cup S'_2, \ldots, S_n \cup S'_n)$.

- Suma zbiorów, które zakładamy, że nie mają wspólnych elementów będzie oznaczana przez $\uplus$. Innymi słowy $A \uplus B = \{x \mid x \in A \cup B\}$ $A \cap B = \emptyset$. Jeżeli operacja sumy nie jest poprzedzona operatorem przypisania $(=)$ to należy rozumieć, że wynik sumowania jest przypisywany do zmiennej reprezentowanej przez lewy argument sumy.

$$(S_1 = S_1 \uplus S_2) \equiv S_1 \uplus S_2$$

- Różnicą zbiorów A i B jest zbiór $C = A \setminus B$ zdefiniowany jako:
  $C = \{x \mid x \in A \wedge x \notin B\}$.

- Iloczyn kartezjański zbiorów A i B to zbiór $C = A \times B$ zdefiniowany jako:
  $A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$.

Będą również stosowane następujące oznaczenia, mające na celu poinformowanie o cechach zbioru:

- $A \subseteq B$ - oznacza, że zbiór A jest podzbiorem właściwym zbioru B, czyli wszystkie elementy zbioru A zawierają się w zbiorze B. Obydwa zbiory nie mogą być identyczne.

- $A \subset B$ - oznacza, że zbiór A jest podzbiorem zbioru B. Różnica względem podzbioru właściwego polega na tym, że obydwa zbiory mogą być sobie równe.

- Liczbę elementów zbioru A, czyli jego liczebność, oznaczymy przez: $\overline{A}$. Przykładowo dla zbioru $A = \{a, b, c\}$ jego liczebność to $\overline{A} = 3$.

W literaturze poświęconej dwugrafom (zdefiniowanym w dalszej części tej pracy) wykorzystywane są zbiory porządkowe (ang. *ordinals*) składające się z

liczb naturalnych od 0 do pewnej wartości. Liczebność takich zbiorów jednocześnie jest wykorzystywana jako symbol je oznaczający. Zbiory porządkowe oznaczane będą małą literą m, n lub k. Dla zachowania zgodności z aktualną literaturą w tej pracy będzie stosowana ta notacja. Będą również wykorzystywane następujące operacje na zbiorach porządkowych:

- $n + m$ - wynikiem jest zbiór porządkowy, którego elementami są liczby od *0* do $n + m - 1$.

- $n-m$ - operacja ta jest równoważna różnicy zbiorów $n \backslash m$ i jej wynikiem jest zbiór, którego elementami są liczby całkowite od m do *n-1*. W przypadku tej operacji zakładamy, że $n - m \geq 1$.

Dla każdego zbioru X istnieje zbiór, którego elementami są wszystkie jego podzbiory. Taki zbiór określany jest zbiorem potęgowym zbioru X i oznaczany będzie 2^X.

Dla dowolnych dwóch zbiorów X oraz Y funkcja (albo przekształcenie[1]) $f : X \to Y$ przypisuje dokładnie jeden element zbioru Y każdemu elementowi zbioru X. Dla tak zdefiniowanej funkcji zbiór X jest jej dziedziną (lub zbiorem argumentów), a Y jest zbiorem, do którego należą wartości funkcji. Ten drugi zbiór należy jednak odróżnić od zbioru wartości funkcji tj. wszystkich elementów y zbioru Y, dla których istnieje $x \in X$ takie, że $f(x) = y$. Zbiór wartości będzie również określany przeciwdziedziną funkcji *f*. W dalszej części pracy będą w zdecydowanej większości rozpatrywane funkcje, które nie przypisują wszystkim elementom zbioru X wartości, a tylko jego podzbiorowi. Takie funkcje są nazywane częściowymi. Określenie to będzie pomijane w większości przypadków i zapis $f : X \to Y$ będzie oznaczał zarówno funkcje „całkowite", jak i częściowe. Dziedziną funkcji częściowej $f : X \to Y$ będziemy określali podzbiór X, dla którego funkcja f jest określona.

Podobnie jak w przypadku zbiorów, funkcje będziemy definiować na kilka sposobów. Pierwszy sposób określa tylko przekształcane zbiory. Przykładem jest wcześniej przedstawiony zapis funkcji $f : X \to Y$, który informuje, że elementy

[1]W tej pracy pojęcia przekształcenia i funkcji będą równoważne. W literaturze można znaleźć pozycje, które zarówno stosują takie ujednolicenie [69], jak i rozróżniają te dwa pojęcia [74].

zbioru X są przekształcane w elementy zbioru Y. Ten sposób definiowania funkcji będzie używany, gdy przekształcenie będzie formułowane opisowo. Drugi sposób definiowania funkcji to rozpatrywanie jej jako zbioru par uporządkowanych, w którym pierwszym elementem każdej pary jest argument funkcji, a drugim jest wartość dla tego argumentu. Ten sposób definiowania będzie używany tylko dla funkcji o skończonych i mało licznych dziedzinach. W przypadku gdy wartość funkcji dla jakiegoś argumentu będzie zależna od innych funkcji, będziemy stosować warunkową formę definicji. Przykładowo funkcja (częściowa) f, która zwraca kwadrat dowolnego argumentu mniejszego od 0, a pierwiastek każdego argumentu większego od zera można zapisać jako:

$$f(x) = \begin{cases} \sqrt{x}\ , & \text{jeżeli} \quad x > 0\ , \\ x^2\ , & \text{jeżeli} \quad x < 0\ . \end{cases}$$

Funkcjonałem [39] nazwiemy funkcję, która jako argument przyjmuje inną funkcję, a której wartości to liczby. W dalszej części pracy będziemy traktować następujące funkcje oraz funkcjonały jako znane:

- $dom : (A \to B) \to A$ - funkcjonał przyjmujący jako argument funkcję, a zwracający jej dziedzinę. W przypadku funkcji częściowej zwracany jest zbiór tylko tych argumentów, dla których jest ona określona.

- $codom$: $(A \to B) \to B$ - funkcjonał przyjmujący jako argument funkcję, a zwracający jej przeciwdziedzinę.

- $\overrightarrow{X}\downarrow (x)$: $X \to \mathbb{N} \quad x \in X$ - funkcja, która przekształca zbiór X do zbioru porządkowego, a następnie zwraca wartość z tego zbioru odpowiadającą elementowi z pierwotnie przekształconego zbioru. Sposób przypisania elementom zbioru X elementów zbioru porządkowego jest bez znaczenia, ale zawsze dla takiego samego zbioru sposób przypisania jest taki sam. Przykładowo zbiór $X = \{1, 777, 21\}$ po przekształceniu w zbiór porządkowy będzie miał postać $m = \{0, 1, 2\}$. Przypuśćmy również, że sposób przypisania elementom zbioru X elementów zbioru m jest następujący: $\{(1, 0), (777, 1), (21, 2)\}$. Dla tak zdefiniowanych elementów przekształcenie $\overrightarrow{X}\big\downarrow (777)$ zwróci wartość *1*.

- min: $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ - funkcja, która zwraca mniejszy z dwóch argumentów podanych na wejściu. Przykładowo $min(7, 4) = 4$.

- max: $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ - funkcja, której wartością jest większy z dwóch argumentów podanych na wejściu. Przykładowo $max(7, 4) = 7$.

Dla dwóch funkcji $f : X \to Y$ i $g : Y \to Z$ ich superpozycja albo złożenie oznaczane przez $g \circ f$ oznacza nową funkcję $h : X \to Z$. W przypadku, gdy funkcja g jest częściowa wynik złożenia zostaje ograniczony do argumentów funkcji f, dla których wartość należy do dziedziny funkcji g. Przykładowo złożenie funkcji $g \circ f$ gdzie:

$$f = \{(1, 2), (2, 3)\} \quad g = \{(3, 7), (4, 8)\}$$

da w rezultacie funkcję $\{(2, 7)\}$.

Zbiór wszystkich funkcji przekształcających elementy zbioru X w elementy zbioru Y będzie zapisywany jako Y^X albo $\{f : X \to Y\}$. Dla dowolnych dwóch funkcji $f_1 : X \to Y$ oraz $f_2 : A \to B$, których dziedziny są rozłączne działanie operatora $\uplus$ należy rozumieć jako:

$$(f_1 \uplus f_2)(x) = \begin{cases} f_1(x)\ , & \text{jeżeli} \quad x \in X\ , \\ f_2(x)\ , & \text{jeżeli} \quad x \in A\ . \end{cases}$$

Jeżeli dowolna z powyższych funkcji jest „pusta” tj. której dziedzina jest zbiorem pustym to operacja $f \uplus \emptyset = \emptyset \uplus f = f$. Suriekcją będzie określana każda funkcja $f : X \to Y$, której przeciwdziedzina jest równa zbiorowi Y. Iniekcją będzie określana każda funkcja $f : X \to Y$, która dla różnych argumentów przyjmuje różne wartości. Bijekcją będzie określana każda funkcja $f : X \to Y$, która jest jednocześnie suriekcją i iniekcją.

Funkcję $i : X \to X$, która przekształca elementy jakiegoś zbioru X w nie same nazwiemy *funkcją tożsamościową* (albo tożsamością). Takie funkcje będą oznaczane i_X z indeksem dolnym wskazującym na zbiór, który przekształcają.

Funkcja przekształcająca funkcję zdaniową w wartość liczbową ze zbioru $\{0, 1\}$ jest określana symbolem Iversona i oznaczana będzie nawiasami kwadratowymi $[\ L\ ]$, gdzie L oznacza przekształcane wyrażenie logiczne. Formalnie można tę

funkcję zdefiniować jako:

$$[\ L\ ] = \begin{cases} 1\ , & \text{jeżeli } L \text{ jest prawdziwe ,} \\ 0 & \text{w przeciwnym wypadku .} \end{cases}$$

W kontekście funkcji operator $\subset$ jest zdefiniowany następująco:

$$f \subset g = \begin{cases} \text{prawda} , & \text{jeżeli } dom(f) \subset dom(g) \wedge \forall x \in dom(f) \quad f(x) = g(x) , \\ \text{fałsz} & \text{w pozostałych przypadkach .} \end{cases}$$

2.2 Grafy

Definicje wprowadzone w tym rozdziale zostały zaadaptowane z [53]. *Grafem* nazwiemy strukturę matematyczną składającą się z wierzchołków i krawędzi. W literaturze można znaleźć różne reprezentacje grafów [53, 50]. W tej pracy przyjęto, że grafem G nazywana będzie trójka $G = (V, E, C)$, gdzie V stanowić będzie zbiór wierzchołków grafu, przez E oznaczany będzie zbiór gałęzi, a $C \subset V \times E \times V$ oznacza relację trójczłonową definiującą połączenia pomiędzy wierzchołkami.

Graf, w którym gałęzie $e \in E$ należące do trójki $(v_1, e, v_2) \in C$ dla $v_1 \neq v_2$ należą jednocześnie do $(v_2, e, v_1) \in C$ określany będzie nieskierowanym. W przypadku odwrotnym tj. gdy dla różnych wierzchołków $v_1 \neq v_2$ żadna gałąź $e \in E$ nie należy jednocześnie do $(v_1, e, v_2) \in C$ oraz $(v_2, e, v_1) \in C$ będziemy mówić o grafie skierowanym. Gałęzie w grafie nieskierowanym będą określane krawędziami, a w grafach skierowanych łukami.

W tej pracy nie będą rozpatrywane grafy, w których mogą jednocześnie wystąpić krawędzie oraz łuki.

W przypadku, gdy pomiędzy dwoma różnymi wierzchołkami może wystąpić więcej niż jedna gałąź mamy do czynienia z *multigrafem.*

Podgrafem grafu $G = (V, E, C)$ jest każdy graf $G' = (V', E', C')$ spełniający poniższe ograniczenia:

- $V' \subset V$,

- $E' \subset E$,
- $C' \subset C$,
- $\forall x, y \in V' \quad (x, e, y) \in C \Leftrightarrow e \in E' \wedge (x, e, y) \in C'$.

Intuicyjnie można rozumieć podgraf jako dowolny podzbiór wierzchołków grafu pierwotnego wraz ze wszystkimi gałęziami pomiędzy tymi wierzchołkami, jakie istniały w grafie pierwotnym.

Marszrutą (skierowaną) w grafie $G = (V, E, C)$ określany jest ciąg przemienny wierzchołków i gałęzi:

$$(v_0, u_1, v_1, u_2, \ldots, u_l, v_l) \quad v_i \in V \; u_j \in E$$

Każda marszruta musi spełniać warunek:

$$\forall i \in \{1, \ldots, l\} \quad (v_{i-1}, u_i, v_i) \in C$$

Długością marszruty (czasami określaną w tej pracy liczbą jej kroków) określana jest liczba gałęzi w ciągu ją definiującym.

Graf spójny to taki, w którym każde dwa wierzchołki można połączyć marszrutą.

Składową spójności grafu nazywamy podgraf będący grafem spójnym, w którym zbiór wierzchołków nie jest podzbiorem właściwym żadnego innego podzbioru wierzchołków, tworzących składową spójności. Składowe spójności można intuicyjnie rozumieć jako podgrafy, w których wszystkie wierzchołki można połączyć marszrutą, a jednocześnie nie można dodać żadnego dodatkowego wierzchołka z grafu pierwotnego. Liczbę składowych spójności grafu G będziemy oznaczać przez $c(G)$.

Liczba cyklomatyczna grafu to suma liczby krawędzi w grafie oraz liczby składowych spójności tego grafu pomniejszona o liczbę wierzchołków. Można ją zapisać równaniem $\alpha(G) = \overline{E} + c(G) - \overline{V}$.

Lasem nazywamy graf, którego liczba cyklomatyczna jest równa 0.

Drzewem nazywamy las, który ma jedną składową spójności.

Ostatnim wprowadzanym w tym podrozdziale pojęciem będzie definicja hipergrafu. *Hipergraf* to uogólnienie grafu, w którym gałęzie mogą łączyć dowolną liczbę wierzchołków.

2.3 Dwugrafy

W związku z tym, że w języku polskim nie została jeszcze ustalona terminologia związana z tematyką poruszaną w tym rozdziale kluczowe pojęcia pochodzące z obcojęzycznej literatury będą wprowadzane i zestawione z oryginalnym nazewnictwem w języku angielskim.

Dwugrafy (ang. *bigraphs*) to struktury matematyczne opracowane przez Robina Milnera [62], umożliwiające modelowanie systemów, w których istotną rolę odgrywa zarówno rozmieszczenie elementów (ang. *placing*) jak i połączenia między nimi (ang. *linking*). Każdy dwugraf składa się z grafu rozmieszczenia (ang. *place graph*) oraz (hiper)grafu połączeń (ang. *link graph*). Do cech charakterystycznych dwugrafów można zaliczyć:

- Formalne zdefiniowanie warunków istnienia wzorców dwugrafowych. Umożliwiają one znacznie bardziej zaawansowane weryfikowanie modelu pod kątem poszukiwanych cech systemu niż np. podgrafy.
- Dekompozycja dwugrafów zdefiniowana jest w taki sposób, że możliwe jest zarówno późniejsze zrekonstruowanie dwugrafu pierwotnego, jak i zmodyfikowanie jego fragmentów.
- Możliwość modelowania dynamiki systemów przy użyciu wyłącznie notacji dwugrafowej.

Aktualnie istnieje kilka rozszerzeń formalizmu wprowadzonego przez Milnera, umożliwiających zastosowanie dwugrafów do większej grupy systemów. Jako przykładowe można wskazać rozszerzenie umożliwiające uwzględnienie prawdopodobieństwa zjawisk, które mogą zajść w systemie [54] oraz zmieniające relacje hierarchiczne pomiędzy elementami z funkcji w relacje binarne [77]. Drugie ze wskazanych rozszerzeń umożliwia m.in. modelowanie sytuacji, w których elementy modelowanego systemu znajdują się w kilku miejscach jednocześnie. Przy-

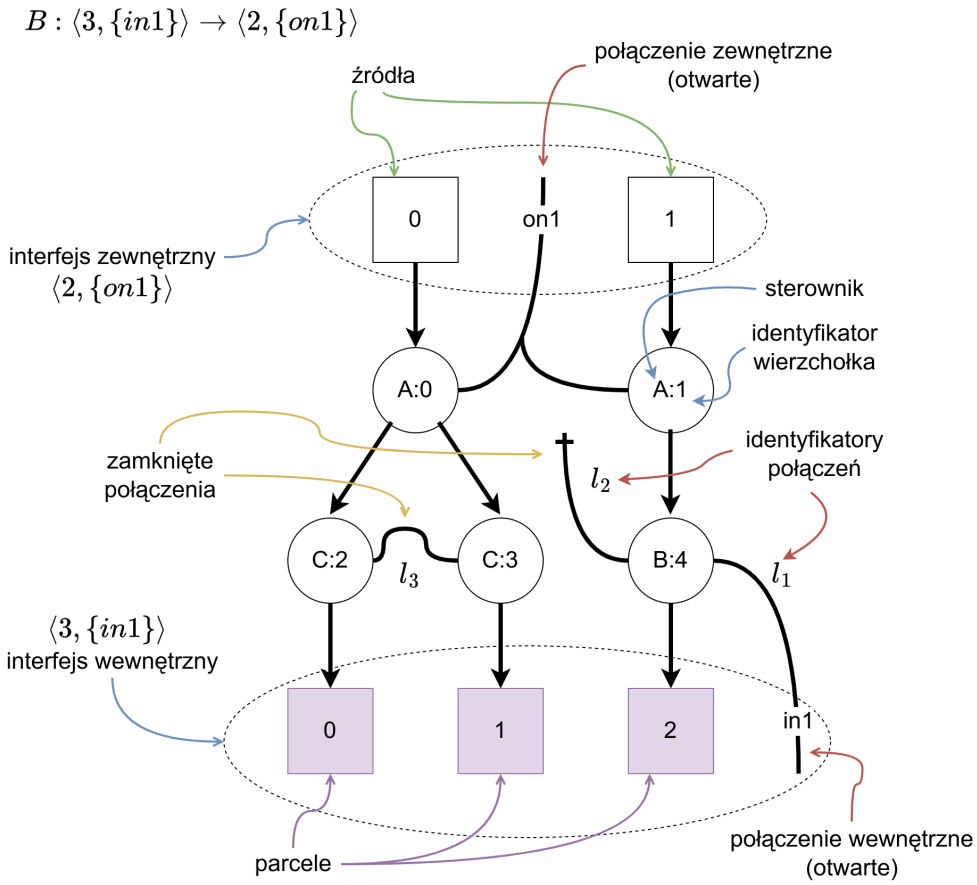
kładem praktycznym zastosowania tego rozszerzenia może być modelowanie sieci bezprzewodowych, z których kilka może obejmować jednocześnie ten sam obszar.

Bardzo przystępne wprowadzenie do praktycznego zastosowania dwugrafów wraz z rozszerzeniami zostało przedstawione w [9]. W tej pracy nie będą uwzględniane rozszerzenia dwugrafów wykraczające poza te zaproponowane w [62].

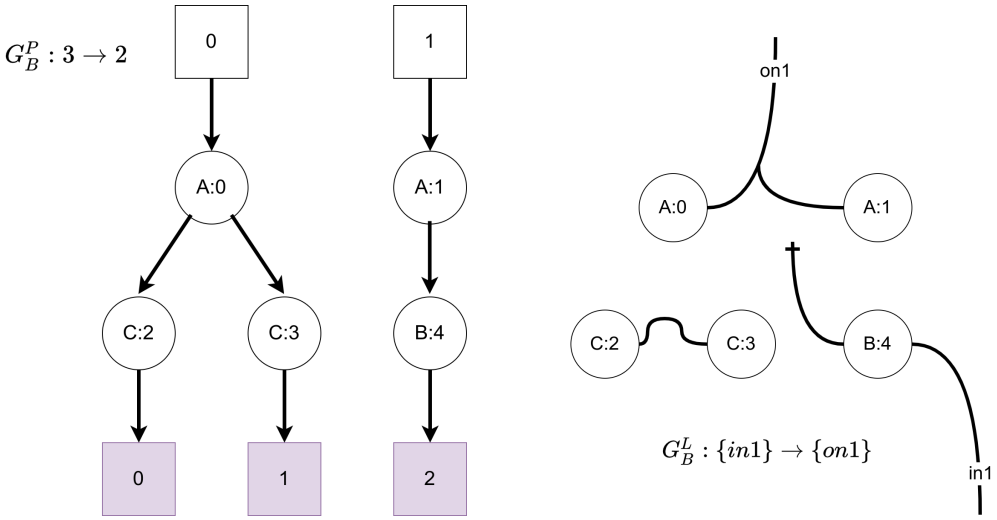
Ten podrozdział poświęcony jest wyłącznie modelowaniu statycznych aspektów systemów przy użyciu dwugrafów. Dynamika systemu modelowanego formalizmem dwugrafowym jest przedmiotem następnego podrozdziału.

Elementy świata rzeczywistego są reprezentowane na dwugrafie przy użyciu wierzchołków. Dwugrafy umożliwiają modelowanie binarnych relacji hierarchicznych pomiędzy obiektami świata rzeczywistego oraz wieloelementowych relacji, w których wszystkie elementy są równoważne.
Pierwszy ze wspomnianych typów relacji umożliwia przedstawienie takich zależności pomiędzy elementami jak np. posiadanie jednego przez drugi, albo obecność osób na poszczególnych piętrach budynku. Ten typ relacji w dwugrafach ma charakter funkcji, tzn. każdy element może mieć maksymalnie jeden element nadrzędny. Relacje hierarchiczne pomiędzy elementami są zawarte w *grafie rozmieszczenia* (ang. *place graph*), będącego składową dwugrafu. Graf rozmieszczenia można rozpatrywać jako zbiór drzew w rozumieniu teorii grafów.
Drugi typ relacji pomiędzy elementami dwugrafu nie wyróżnia w nich elementu nadrzędnego. Ten typ powiązań może zostać wykorzystany do przedstawienia np. zdolności do komunikacji w sieci, w której wszyscy mogą komunikować się z pozostałymi członkami sieci albo do modelowania możliwych sposobów przejścia pomiędzy obszarami, w przypadkach gdzie używając tej samej drogi można dostać się do kilku obszarów. Relacje niehierarchiczne pomiędzy elementami są definiowane na *grafie połączeń* (ang. *link graph*). Graf połączeń jest hipergrafem.

Każdy wierzchołek w dwugrafie ma przypisany typ określany sterownikiem (ang. *control*). Z każdym sterownikiem powiązana jest liczba *portów*, która określa z iloma połączeniami (ang. *links*) wierzchołek danego typu może być powiązany. Sygnatura dwugrafu (ang. *signature*) to para składająca się ze zbioru możliwych sterowników (oznaczanego jako $\mathcal{K}$) wraz z funkcją przypisującą każ-



(a) Przykładowy dwugraf



(b) Graf rozmieszczenia będący składową dwugrafu z rysunku 2.1a.

(c) Graf połączeń będący składową dwugrafu z rysunku 2.1a.

Rysunek 2.1: Przykład dwugrafu z podziałem na składowe.

demu z nich liczbę portów. Połączenia są elementami grafu połaczeń, które można podzielić na zamknięte (ang. *closed links*) oraz otwarte (ang. *open links*). Połączeniami zamkniętymi są hiperkrawędzie, przy czym mogą one „łączyć" jeden wierzchołek nie będąc jednocześnie pętlą. Połączenia otwarte są *stronami*[2] grafu połączeń. Rozróżniana jest strona zewnętrzna (ang. *outer face*) oraz wewnętrzna (ang. *inner face*) grafu połączeń, jak i grafu rozmieszczenia. Elementy strony zewnętrznej grafu połączeń określane będą połączeniami zewnętrznymi, a elementy strony wewnętrznej połączeniami wewnętrznymi. Odpowiednikiem w grafie rozmieszczenia strony zewnętrznej będą *źródła* (ang. *roots*), a wewnętrznej *parcele* (ang. *sites*). Powiązania pomiędzy elementami grafu rozmieszczenia są tylko jednego rodzaju i są łukami w rozumieniu teorii grafów. Mogą one łączyć zarówno źródła z wierzchołkami, wierzchołki z (innymi) wierzchołkami, wierzchołki z parcelami, a także źródła z parcelami.
Dwugraf nad określoną sygnaturą składa się z grafu rozmieszczenia oraz grafu połączeń. Przykład dwugrafu wraz z podziałem na graf rozmieszczenia oraz połączeń zamieszczono na rysunku 2.1. Dla każdego dwugrafu można określić interfejs (ang. *interface*) zewnętrzny oraz wewnętrzny, na który składa się strona odpowiednio zewnętrzna oraz wewnętrzna grafu połączeń i rozmieszczenia.

W tej pracy są używane głównie dwugrafy konkretne, w których do każdego wierzchołka przypisany jest unikalny identyfikator. Inaczej jest w przypadku dwugrafów abstrakcyjnych, w których takiej identyfikacji nie ma. Jeżeli nie będzie zaznaczone, że jest inaczej to w tej pracy przez dwugrafy należy rozumieć dwugrafy konkretne.

Formalnie dwugraf konkretny jest zdefiniowany jako:

$$B = (V_B, ctrl_B, G_B^P, G_B^L) : I \to O$$

gdzie:

- V_B - zbiór identyfikatorów wierzchołków,

[2]W terminologii angielskiej połączenia otwarte stanowią *inner and outer names of the link graph*[62]. Ze względu na brak odpowiednika słowa *names*, który oddawałby funkcję pełnioną przez te elementy zdecydowano się na użycie określenia *strony* (ang. *faces*), które jest używane zarówno w kontekście grafów połączeń jak i rozmieszczeń.

- $ctrl_B : V_B \to \mathcal{K}$ - funkcja przypisująca wierzchołkom typ sterownika,
- $G_B^P = (V_B, ctrl_B, prnt_B) : m \to n$ - graf rozmieszczenia będący składową dwugafu B, którego nowymi elementami składowymi są:
  - m - liczba parceli, która pełni jednocześnie rolę zbioru porządkowego;
  - n - liczba źródeł, która tak samo jak powyżej pełni rolę zbioru porządkowego;
  - $prnt_B : m \uplus V_B \to V_B \uplus n$ - funkcja określająca hierarchiczną zależność pomiędzy wierzchołkami, parcelami i źródłami. Definiuje zbiór drzew;
- $G_B^L = (V_B, E_B, ctrl_B, link_B) : X \to Y$ - graf połączeń będący składową dwugafu B, którego nowymi elementami składowymi są:
  - E_B - zbiór identyfikatorów połączeń (hiperkrawędzi);
  - X - zbiór połączeń otwartych wewnętrznych;
  - Y - zbiór połączeń otwartych zewnętrznych;
  - $link_B : X \uplus P_B \to E_B \uplus Y$ - funkcja łącząca połączenia otwarte wewnętrzne oraz porty wierzchołków z połączeniami zamkniętymi lub połączeniami otwartymi zewnętrznymi.
    Zbiór P_B jest zdefiniowany jako $P_B = \{(v, i) \mid i \in \mathbb{N}_+ \; v \in V_B\}$, a jego elementy (v, i) oznaczają i-ty port wierzchołka v;
- $I = \langle m, X \rangle$ i $O = \langle n, Y \rangle$ są odpowiednio wewnętrznym i zewnętrznym interfejsem dwugrafu B.

Korzystając z powyższych definicji, dwugraf przedstawiony w postaci graficznej na rysunku 2.1 został dalej formalnie zdefiniowany. Sygnatura tego dwugrafu jest następująca:

$$(\mathcal{K} = \{\text{A,B,C}\}, \{(A, 1), (B, 2), (C, 1)\})$$

Część wspólna dla grafu rozmieszczenia, jak i grafu połączeń dwugrafu z rysunku 2.1 formalnie ma postać:

$$V_B = \{0, 1, 2, 3, 4\}$$

$$ctrl_B = \{(0, A), (1, A), (2, C), (3, C), (4, B)\}$$

Funkcja $prnt_B$, będąca częścią grafu rozmieszczenia dwugrafu z rysunku 2.1 formalnie ma postać:

$$prnt_B = \{(0, \boxed{0}), (1, \boxed{1}), (2, 0), (3, 0), (4, 1), (\boxed{0}, 2), (\boxed{1}, 3), (\boxed{2}, 4)\}$$

Kwadraty z szarym wypełnieniem reprezentują parcele, a kwadraty białe odpowiadają źródłom. Zapis ten ma celu ułatwić rozróżnienie między identyfikatorami wierzchołków, a identyfikatorami stron grafu rozmieszczenia. Według oznaczeń wprowadzonych wcześniej wartość m odpowiadająca liczbie parceli jest równa 3. Interpretując m jako zbiór porządkowy będzie on zdefiniowany jako $m = \{\boxed{0}, \boxed{1}, \boxed{2}\}$. Liczba źródeł oznaczana przez n jest równa 2. Zbiór identyfikatorów połączeń E_B grafu połączeń dwugrafu z rysunku 2.1 został zdefiniowany jako:

$$E_B = \{l_1, l_2, l_3\}$$

Strony tego grafu połączeń zawierają zarówno połączenia wewnętrzne, jak i zewnętrzne. Zbiór połączeń wewnętrznych ma postać:

$$X = \{\text{in1}\}$$

Z kolei, zbiór połączeń zewnętrznych został zdefiniowany jako:

$$Y = \{\text{on1}\}$$

Zbiór portów P_B omawianego grafu połączeń przypisuje indeksy portów wszystkim wierzchołkom, zgodnie z ich sterownikami oraz sygnaturą dwugrafu.

$$P_B = \{(0, 1), (1, 1), (2, 1), (3, 1), (4, 1,), (4, 2)\}$$

Ostatnim elementem potrzebnym do formalnego zdefiniowania dwugrafu z rysunku 2.1 jest funkcja $link_B$, która ma postać:

$$link_B = \{(in1, l_1), ((4, 1), l_1), ((4, 2), l_2), ((2, 1), l_3), ((3, 1), l_3), ((0, 1), on1), ((1, 1), on1)\}$$

Tam gdzie to możliwe, pomijane będą elementy składowe dwugrafu i jego de-

finicja ograniczy się do nazwy (B_1, B_2 itp.) oraz do jego interfejsów. Przykładem takiej skróconej definicji dwugrafu będzie $B_x : \langle 7, \emptyset \rangle \to \langle 0, \{x, y, z\} \rangle$.

Podstawa dwugrafu B (ang. *bigraph support*) to zbiór $|B| = V_B \uplus E_B$. Podstawą grafu rozmieszczenia jest sam zbiór $|G_B^P| = V_B$, a podstawa grafu połączeń równa podstawie całego dwugrafu tj. $|G_B^L| = V_B \uplus E_B$.

Dwa dwugrafy B_1 i B_2 o takich samych interfejsach $\langle m, X \rangle \to \langle n, Y \rangle$ są *równoważne z dokładnością do podstawy* (ang. *support equivalent*), jeżeli istnieje para funkcji $\rho_V : V_{B_1} \to V_{B_2}$ i $\rho_E : E_{B_1} \to E_{B_2}$ spełniająca poniższe warunki:

- ρ_V zachowuje strukturę dwugrafu B_1 tzn. $ctrl_{B_2} \circ \rho_V = ctrl_{B_1}$.
- Złożenie obydwu funkcji ρ i funkcji definiujących struktury grafów rozmieszczenia oraz połączeń jest przemienne, tzn.:
  $prnt_{B_2} \circ (i_m \uplus \rho_V) = (i_n \uplus \rho_V) \circ prnt_{B_1}$,
  $link_{B_2} \circ (i_X \uplus \rho_P) = (i_Y \uplus \rho_E) \circ link_{B_1}$.
  Funkcja $\rho_P : P_{B_1} \to P_{B_2}$ przypisuje portom dwugrafu B_1 porty dwugrafu B_2 tzn. $\rho_P((v, i)) = (\rho_V(v), i)$.

Para funkcji (ρ_V, ρ_E) będzie określana *przekształceniem podstawy* (ang. *support translation*) i oznaczana jako $\rho_{B_1 \to B_2}$.

Dwugraf jest zwięzły (ang. *lean*), jeżeli nie ma połączeń zamkniętych nieprzyłączonych do żadnych portów. Formalnie dwugraf B jest zwięzły gdy:

$$\forall e \in E_B \; \exists p \in P_B : \quad link_B(p) = e$$

Dwugrafy B_1, B_2 są *równoważne z dokładnością do zwięzłych podstaw* (ang. *lean-support equivalent*), jeżeli po usunięciu z obydwu wszystkich zamkniętych połączeń nieprzyłączonych do żadnego portu są równoważne z dokładnością do podstaw.

Dwugrafy określimy rozłącznymi (ang. *disjoint*), jeżeli ich podstawy nie mają części wspólnej. Formalnie dwa dwugrafy B_1, B_2 są rozłączne, jeżeli:

$$V_{B_1} \cap V_{B_2} = \emptyset \wedge E_{B_1} \cap E_{B_2} = \emptyset$$

Dwugraf abstrakcyjny (ang. *abstract bigraph*) to zbiór dwugrafów konkret-

nych, które są równoważne z dokładnością do zwięzłych podstaw. Dwa dwugrafy B_1 i B_2 należące do tego samego dwugrafu abstrakcyjnego oznaczymy przez $B_1 \cong B_2$.

Dwa dwugrafy konkretne są sobie równoważne, jeżeli należą do tego samego zbioru oznaczającego dwugraf abstrakcyjny[3].

Dwugrafy mogą być zarówno częścią większej całości jak i stanowić szablon do wypełnienia przez inne dwugrafy. Można dokonać połączenia dwugrafów rozłącznych tylko, jeżeli mają kompatybilne interfejsy. Na potrzeby dalszych części pracy zdefiniowane zostaną dwie operacje łączące dwugrafy tj. kompozycja (ang. *composition*) oraz zestawienie (ang. *juxtaposition*).

Kompozycja rozłącznych dwugrafów

$$B_1 : \langle n, X \rangle \to \langle m, Y \rangle \quad B_2 : \langle k, Z \rangle \to \langle n, X \rangle$$

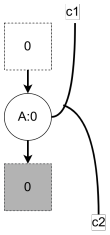
zapisana jako $B_1 \circ B_2$ daje w wyniku dwugraf B_3:

$$B_3 : \langle k, Z \rangle \to \langle m, Y \rangle$$

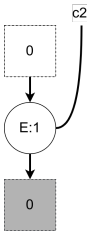
którego elementy składowe są definiowane następująco:

- $V_{B_3} = V_{B_1} \uplus V_{B_2}$,
- $ctrl_{B_3} = ctrl_{B_1} \uplus ctrl_{B_2}$,
- $G^P_{B_3} = (V_{B_3}, ctrl_{B_3}, prnt_{B_3}) : k \to m$,
  - V_{B_3} oraz $ctrl_{B_3}$ jak zdefiniowano wyżej;
  - $prnt_{B_3}(w) = \begin{cases} prnt_{B_2}(w) , & \text{jeżeli} \quad w \in k \uplus V_{B_2} \wedge prnt_{B_2}(w) \in V_{B_2} , \\ prnt_{B_1}(j) , & \text{jeżeli} \quad w \in k \uplus V_{B_2} \wedge prnt_{B_2}(w) = j \in n , \\ prnt_{B_1}(w) , & \text{jeżeli} \quad w \in V_{B_1} . \end{cases}$
- $G^L_{B_3} = (V_{B_3}, E_{B_3}, ctrl_{B_3}, link_{B_3}) : Z \to Y$
  - V_{B_3} i $ctrl_{B_3}$ jak zdefiniowano wyżej;

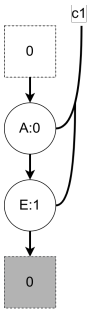
[3]W [62] dwugrafy abstrakcyjne to klasy równoważności morfizmów w symetrycznej częściowo monoidalnej kategorii (ang. *symmetric partial monoidal category*). Określenie „częściowa" dotyczy operacji produktu tensorowego $\otimes$, która w takich kategoriach nie jest zdefiniowana dla wszystkich obiektów. W tej pracy perspektywa teorii kategorii nie jest uwzględniana, stąd różnica w definicjach.



(a) Dwugraf $B_1 : \langle 1, \{c2\}\rangle \to \langle 1, \{c1\}\rangle$ będący lewą składową kompozycji.



(b) Dwugraf $B_2 : \langle 1, \emptyset\rangle \to \langle 1, \{c2\}\rangle$ będący prawą składową kompozycji.



(c) Wynik kompozycji $B_1 \circ B_2$ będący dwugrafem $B_3 : \langle 1, \emptyset\rangle \to \langle 1, \{c1\}\rangle$.

Rysunek 2.2: Przykład operacji kompozycji dwugrafów.

– $E_{B_3} = E_{B_1} \uplus E_{B_2}$;

– $link_{B_3}(q) = \begin{cases} link_{B_2}(q)\ , & \text{jeżeli} \quad q \in Z \uplus P_{B_2} \wedge link_{B_2}(q) \in E_{B_2}\ , \\ link_{B_1}(y)\ , & \text{jeżeli} \quad q \in Z \uplus P_{B_2} \wedge link_{B_2}(q) = y \in X\ , \\ link_{B_1}(q)\ , & \text{jeżeli} \quad q \in P_{B_1}\ . \end{cases}$

Przykład graficzny operacji kompozycji został przedstawiony na rysunku 2.2. Przedstawiono tam kompozycję dwugrafów:

$$B_1 : \langle 1, \{c2\}\rangle \to \langle 1, \{c1\}\rangle \quad B_2 : \langle 1, \emptyset\rangle \to \langle 1, \{c2\}\rangle$$

Zgodnie z definicją przedstawioną powyżej kompozycja $B_1 \circ B_2$ powyższych dwugrafów daje w wyniku dwugraf B_3:

$$B_3 : \langle 1, \emptyset\rangle \to \langle 1, \{c1\}\rangle$$

Drugą podstawową operacją na dwugrafach jest ich *zestawienie*.
Operacja zestawienia rozłącznych dwugrafów:

$$B_1 : \langle m_1, X\rangle \to \langle n_1, Y\rangle \quad B_2 : \langle m_2, Z\rangle \to \langle n_2, W\rangle$$

zapisywana jako $B_1 \otimes B_2$ daje w wyniku dwugraf B_3:

$$B_3 : \langle m_1 + m_2, X \uplus Z \rangle \to \langle n_1 + n_2, Y \uplus W \rangle$$

Elementy składowe dwugrafu, będącego wynikiem zestawienia, są definiowane następująco:

- $V_{B_3} = V_{B_1} \uplus V_{B_2}$,
- $E_{B_3} = E_{B_1} \uplus E_{B_2}$,
- $ctrl_{B_3} = ctrl_{B_1} \uplus ctrl_{B_2}$,
- $prnt_{B_3}(w) = \begin{cases} prnt_{B_1}(w) , & \text{jeżeli} \quad w \in m_1 \uplus V_{B_1} , \\ prnt'_{B_2}(w) , & \text{jeżeli} \quad w \in m'_2 \uplus V_{B_2} ; \end{cases}$

  Należy zwrócić uwagę na zmodyfikowaną funkcję $prnt'_{B_2}$ oraz zbiór m'_2, które są definiowane następująco:

  - $m'_2 = ((m_1 + m_2) - m_1)$ - elementami tego zbioru porządkowego są liczby całkowite od m_1 do $m_2 - 1$.
  - Funkcja $prnt'_{B_2}$ została zdefiniowana w [62] jako:

    $$prnt'_{B_2}(m_1 + i) = n_1 + j \quad \text{jeżeli} \quad prnt_{B_2}(i) = j$$

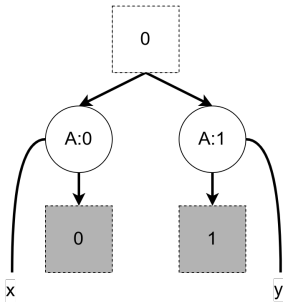
    W tej pracy ma ona następującą postać:

    $$prnt'_{B_2}(w) = \begin{cases} prnt_{B_2}(w) , & \text{jeżeli} \quad w \in m'_2 \wedge prnt_{B_2}(w) \in V_{B_2} , \\ n_1 + a , & \text{jeżeli} \quad w \in m'_2 \wedge prnt_{B_2}(w) = a \in n_2 , \\ prnt_{B_2}(w) , & \text{jeżeli} \quad w \in V_{B_2} \wedge prnt_{B_2}(w) \in V_{B_2} , \\ n_1 + b , & \text{jeżeli} \quad w \in V_{B_2} \wedge prnt_{B_2}(w) = b \in n_2 . \end{cases}$$

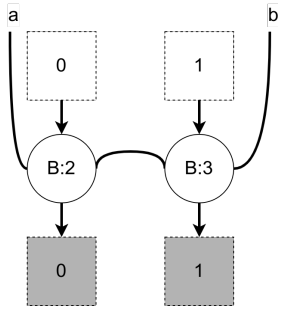
- $link_{B_3} = link_{B_1} \uplus link_{B_2}$.

Przykład operacji zestawienia dwugrafów został przedstawiony na rysunku 2.3. Dokonano tam zestawienia $B_1 \otimes B_2$ dwugrafów:

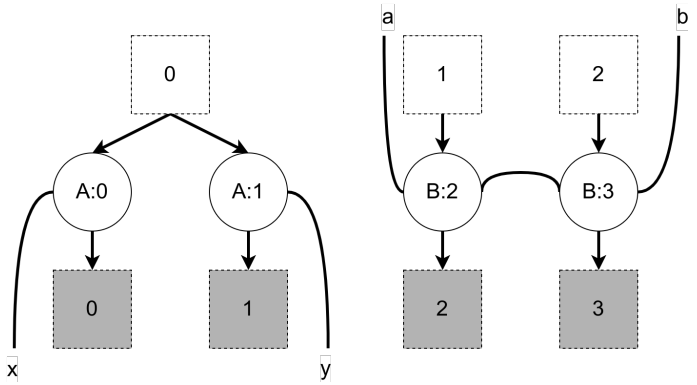
$$B_1 : \langle 2, \{x, y\} \rangle \to \langle 1, \emptyset \rangle \quad B_2 : \langle 2, \emptyset \rangle \to \langle 2, \{a, b\} \rangle$$



(a) Lewa składowa operacji zestawienia. $B_1 : \langle 2, \{x, y\}\rangle \to \langle 1, \emptyset\rangle$



(b) Prawa składowa operacji zestawienia. $B_2 : \langle 2, \emptyset\rangle \to \langle 2, \{a, b\}\rangle$



(c) Wynik zestawienia $B_1 \otimes B_2 = B_3 : \langle 4, \{x, y\}\rangle \to \langle 3, \{a, b\}\rangle$.

Rysunek 2.3: Przykład operacji zestawienia dwóch dwugrafów.

Wynikiem powyższej operacji będzie dwugraf B_3:

$$B_3 : \langle 4, \{x, y\}\rangle \to \langle 3, \{a, b\}\rangle$$

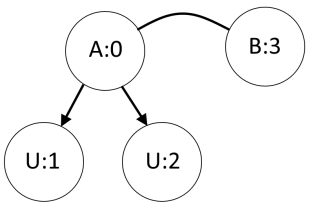
Potrzeba również wspomnieć o *tożsamościach* (ang. *identities*) tj. bezwierzchołkowych dwugrafach, które nie zmieniają dwugrafu z którym są poddawane operacji kompozycji. Tożsamościowym grafem rozmieszczenia nad interfejsem m jest graf rozmieszczenia:

$$id_m = (\emptyset, \emptyset, i_m) : m \to m$$

Tożsamościowym grafem połączenia nad interfejsem X jest graf połączeń:

$$id_X = (\emptyset, \emptyset, \emptyset, i_X) : X \to X$$

Tożsamością dla dwugrafu $B : \langle m, X\rangle \to \langle n, Y\rangle$ nad interfejsem wewnętrznym



Rysunek 2.4: Dwugraf wejściowy do przykładu wprowadzającego

$\langle m, X\rangle$ jest dwugraf:

$$id_{\langle m,X\rangle} = (\emptyset, \emptyset, id_m, id_X)$$

Taka tożsamość nie zmieni dwugrafu B w przypadku kompozycji $B \circ id_{\langle m,X\rangle}$. Analogicznie tożsamość nad interfejsem zewnętrznym $id_{\langle n,Y\rangle}$ nie zmieni dwugrafu B w przypadku kompozycji $id_{\langle n,Y\rangle} \circ B$.

Elementem neutralnym operacji zestawienia dla dowolnego dwugrafu jest dwugraf $id_{\langle 0,\emptyset\rangle} = \langle 0, \emptyset\rangle \to \langle 0, \emptyset\rangle$.

Dotychczas wprowadzone pojęcia umożliwiają zdefiniowanie warunku, którego spełnienie pozwala stwierdzić, że jeden dwugraf występuje (ang. *occurs*) w drugim. Dwugraf B_p znajduje się w dwugrafie B_t jeżeli istnieją takie dwa dwugrafy B_1, B_2 oraz interfejs $I = \langle n, X\rangle$, dla których poniższe równanie jest spełnione:

$$B_t = B_1 \circ (B'_p \otimes id_I) \circ B_2$$

Dwugraf B'_p jest równoważny z dokładnością do zwięzłej podstawy dwugrafowi B_p.

W dalszej części pracy dwugraf B_p będzie określany wzorcem, a B_t dwugrafem wejściowym, w którym jest szukane wystąpienie wzorca. Zbiory dwugrafów wchodzących w skład dekompozycji dwugrafu wejściowego do postaci powyższego równania są określane różnymi wystąpieniami wzorca.

Dwa wystąpienia wzorca są sobie równoważne tylko wtedy, gdy są identyczne lub różnią się:

- „Numeracją" przyłączanych parceli dwugrafu B_1 lub źródeł dwugrafu B_2.
- Zbiorem połączeń wewnętrznych B_1 lub połączeń zewnętrznych B_2.

Dwa wystąpienia wzorca dwugrafowego różniące się obydwiema powyższymi ce-

chami nadal będą sobie równoważne. Przykład graficzny różnych wystąpień tego samego wzorca dwugrafowego zamieszczono na rysunku 2.5.

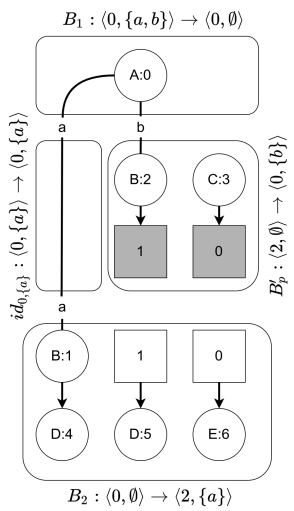
Wzorce dwugrafowe mają charakter „minimalnego satysfakcjonującego zjawiska". Oznacza to, że można definiować przy ich użyciu sytuacje takie, jak: *minimum trzy obiekty typu A* lub *dokładnie trzy obiekty typu A*. Niemożliwym jest jednak przy użyciu jednego wzorca zdefiniowanie wszystkich sytuacji typu *co najwyżej trzy obiekty typu A*.

Język dwugrafów może zostać wykorzystany do przedstawienia elementów świata rzeczywistego, które są istotne z punktu widzenia zadania postawionego przed rojem. Dalej zamieszczono kilka wskazówek, w jaki sposób elementy notacji dwugrafowej mogą przedstawić elementy świata rzeczywistego, którymi kierowano się konstruując przykłady w dalszej części rozprawy.

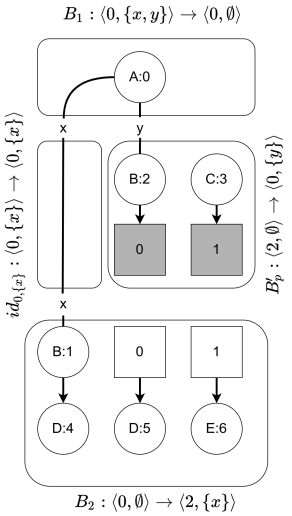
Sterowniki reprezentują typy elementów świata rzeczywistego. Można zaliczyć do nich drony, obszary na mapie, przedmioty, a także niematerialne elementy, jak np. reprezentacja stanu wewnętrznego obiektu, reprezentowanego przez inny sterownik.

Wierzchołki reprezentują wystąpienia elementów świata rzeczywistego, które zostały rozróżnione przy użyciu sterowników. Przykładowo dwa drony tego samego typu (o takim samym sterowniku) będą reprezentowane przez dwa różne wierzchołki. Tak samo dwa obszary na mapie, które są od siebie nierozróżnialne, będą reprezentowane przez dwa wierzchołki tego samego typu. Należy tutaj zaznaczyć, że czym innym jest rozróżnienie samych obszarów, a czym innym ich właściwości, takich jak np. przyległości do innych obszarów, czy tego, co się na nich znajduje.

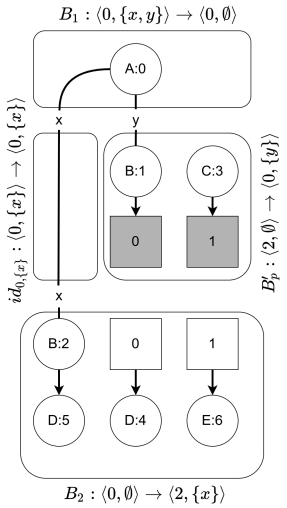
Zależności hierarchiczne przedstawiane na grafie rozmieszczenia pozwalają przedstawić relacje pomiędzy wystąpieniami elementów świata rzeczywistego, w których powiązane elementy nie pełnią równoważnej roli. Najprostszym przykładem zastosowania takiej relacji będzie wszelkiego rodzaju zawieranie pewnych elementów przez (jeden) inny element. Graf rozmieszczenia pozwala również na przedstawienie sytuacji, gdzie pewna grupa elementów (np. drony) jest w jakiś sposób zależna od innego pojedynczego elementu (np. stacji kontrolnej albo operatora).



(a) Przykładowe wystąpienie wzorca dwugrafowego.



(b) Wystąpienie wzorca dwugrafowego równoważne temu z rysunku 2.5a.



(c) Wystąpienie wzorca dwugrafowego nierównoważne tym przedstawionym na rysunkach 2.5a i 2.5b.

Rysunek 2.5: Przykład wystąpień tego samego wzorca dwugrafowego. Wystąpienia wzorca dwugrafowego na rysunkach 2.5a i 2.5b są sobie równoważne, podczas gdy wystąpienie z rysunku 2.5c już nie jest im równoważne.

Zależności niehierarchiczne, w których powiązane elementy są sobie równoważne, przedstawiane są na grafie połączeń. W obszarze rojów dronów, do przykładów takich relacji, można zaliczyć przyległość obszarów albo w ogólności możliwość przemieszczania się pomiędzy nimi. Innym przykładem zależności pomiędzy elementami, które mogą być przedstawione na grafie połączeń, jest możliwość komunikacji pomiędzy poszczególnymi dronami.

Konstruując model systemu z wykorzystaniem dwugrafów, który ma posłużyć do opracowania zachowania dla elementów roju dronów, niezbędnym jest znalezienie stanów systemu, które odzwierciedlają sytuacje docelowe (pożądane). Zamiast definiować wszystkie możliwe stany systemu, które posiadają pożądane cechy, można wykorzystać wzorce dwugrafowe. Jeżeli zostaną znalezione w dwugrafie reprezentującym stan systemu, będą oznaczały, że dany dwugraf reprezentuje stan docelowy. Sposób wykorzystania dotychczas wymienionych elementów grafów rozmieszczenia i połączeń we wzorcach dwugrafowych jest niezmieniony. Należy jednak omówić role jakie pełnią parcele, źródła oraz połączenia, czy w ogólności interfejsy we wzorcach dwugrafowych.

Parcele, będące częścią grafu rozmieszczenia, pozwalają określić, że pewne wierzchołki mogą (ale nie muszą) być w relacji hierarchicznej z innymi, będąc względem nich ńadrzędnymi". Należy podkreślić, że parcele mogą być „wypełnione" źródłami bez następników. W obszarze rojów dronów przykładem zastosowania parceli może być np. potrzeba zdefiniowania obszaru, na którym może znajdować się cokolwiek (włącznie z niczym). Innym przykładem może być konstrukcja wzorca opisującego sytuacje, w których drony są w posiadaniu dowolnej liczby innych obiektów.

Źródła pełnią bardzo podobną rolę do parceli. Służą one zdefiniowaniu sytuacji, w których elementy mogą być (ale nie muszą) w relacji hierarchicznej z dokładnie jednym innym obiektem, będąc względem niego „podrzędnym". Źródła mogą zostać „dołączone" do parceli, nie mających poprzedników. Przykładem wykorzystania źródeł w domenie rojów dronów mogłaby być potrzeba zdefiniowania sytuacji, w której występuje dron, mogący znajdować się w jednym z kilku możliwych typów miejsc. Innym przykładem może być reprezentacja sytuacji, w której tylko niewielki szczegół rzeczywistości decyduje o określeniu stanu jako

docelowy, bez względu na resztę otoczenia.

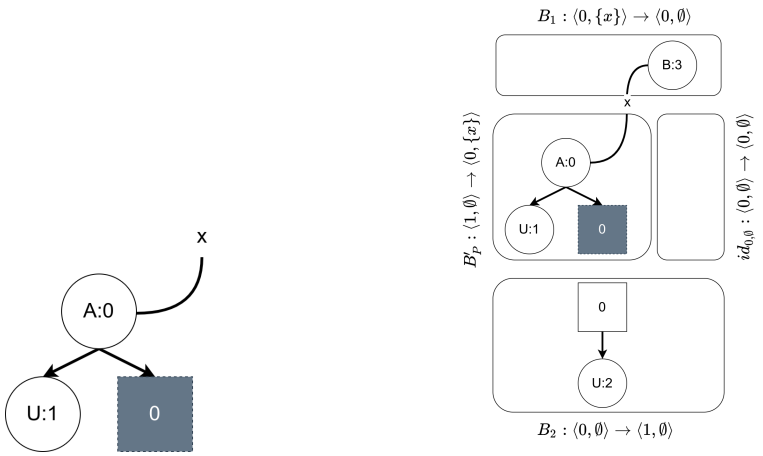
Połączenia otwarte, zarówno wewnętrzne, jak i zewnętrzne pełnią podobną rolę co parcele czy źródła, ale w grafach połączeń.

Wykorzystując połączenia otwarte można przedstawić sytuację, w której pewne elementy są w relacji niehierarchicznej z dowolnymi innymi elementami. Podobnie, jak miało to miejsce w przypadku parceli i źródeł, połączenia mogą zostać „podpięte" do połączeń otwartych wolnych (ang. *idle links*). Oznacza to, że wykorzystanie połączeń otwartych nie wymusza, aby obiekty były połączone z innymi, a jedynie na to pozwala. Przykładem wykorzystania połączeń otwartych w obszarze rojów dronów może być potrzeba określenia sytuacji, w której występuje dron, mogący komunikować się z innym, ale jednocześnie nie jest to wymagane. Innym przykładem zastosowania połączeń otwartych jest sytuacja, gdy spośród kilku połączeń jakiegoś elementu wzorzec dwugrafowy określa tylko ich część. Połączenia otwarte wychodzące z takiego elementu pozwolą na „znalezienie" wzorca w dwugrafach, w których odpowiadające im połączenia mogą być zarówno zamknięte, jak i otwarte.

W ramach przykładu wprowadzającego w tym i następnych rozdziałach rozpatrywany jest system, którego stan początkowy w postaci graficznej został zamieszczony na rysunku 2.4. System ten składa się z dwóch dronów przebywających na jednym z dwóch przyległych do siebie obszarów. Przyległość jest tu reprezentowana przez połączenia w grafie połączeń dwugrafu. Sygnatura dwugrafów reprezentujących stany tego systemu ma postać:

$$(\mathcal{K} = \{U, A, B\}, ar = \{(U : 0), (A : 1), (B : 1)\})$$

Drony są oznaczane sterownikami typu U, a obszary pomiędzy którymi mogą się one przemieszczać sterownikami A i B. Przykładowy wzorzec dwugrafowy umożliwiający wykrycie co najmniej jednego drona na obszarze typu A wraz z jego wykryciem został przedstawiony na rysunku 2.6.



(a) Przykładowy wzorzec dwugrafowy $B_P : \langle 1, \emptyset \rangle \to \langle 0, \{x\} \rangle$ do znalezienia w dwugrafie z rysunku 2.4.

(b) Dekompozycja dwugrafu z rysunku 2.4 zawierająca wzorzec dwugrafowy z rysunku 2.6a.

Rysunek 2.6: Przykład wystąpienia wzorca dwugrafowego

2.4 Dwugrafowe systemy reakcyjne

Celem tego rozdziału jest przedstawienie sposobu zdefiniowania dynamiki systemu z użyciem pojęć języka dwugrafów.

Podstawowym pojęciem tego rozdziału są reguły reakcji, które reprezentują typy zmian, jakie mogą wystąpić w modelowanym systemie. Z ich użyciem można zdefiniować zbiór stanów modelowanego systemu. Dwugrafy reprezentujące stany systemu są uziemione (ang. *ground*) tzn. mają wewnętrzny interfejs postaci $\langle 0, \emptyset \rangle$. Zbiór reguł reakcji wraz ze zbiorem stanów systemu stanowią śledzący dwugrafowy system reakcyjny (ang. *bigraphical reactive system*).

W tej pracy rozpatrywane będą *liniowe parametryczne reguły reakcji* (ang. *linear parametric reaction rules*), zaproponowane w [54], wzbogacone o *śledzenie* (ang. *tracking*) wierzchołków, które zostało opisane[4] w [28]. Zamiast pełnej nazwy tj. śledzących liniowych parametrycznych reguł reakcji będzie używane określenie *reguły reakcji*.

[4]Jest to uproszczona forma śledzenia względem tej przedstawionej w [62], gdzie zdefiniowano śledzenie elementów całej podstawy dwugrafu. W tej pracy używany będzie mechanizm umożliwiający śledzenie tylko wierzchołków dwugrafów.

Śledząca liniowa parametryczna reguła reakcji (dalej reguła reakcji) to trójka:

$$r = (R : \langle m, \emptyset \rangle \to \langle n, X \rangle, R' : \langle m, \emptyset \rangle \to \langle n, X \rangle, \tau)$$

gdzie:

- R - *redeks* reguły reakcji r ,
- R' - *reaktum* reguły reakcji r ,
- τ: $V_{R'} \to V_R$ - funkcja śledząca (ang. *tracking map*). Przypisuje ona wierzchołki redeksu do wierzchołków reaktum. Należy zaznaczyć, że może być to funkcja częściowa tzn. nie każdy wierzchołek reaktum musi być pozostałością (ang. *residue*) redeksu. W przypadku, gdy dla jakiegoś wierzchołka $v \in V_{R'}$ funkcja $\tau(v)$ jest nieokreślona, należy to interpretować, że wierzchołek reprezentuje element nowy, utworzony w wyniku działania reprezentowanego przez regułę reakcji. Natomiast wszystkie elementy $v \in V_R$, które nie należą do przeciwdziedziny funkcji τ oznaczają elementy, które w wyniku czynności r znikają z systemu.

Redeks oraz reaktum muszą być dwugrafami zwięzłymi, ponadto dwugrafy mogące stanowić redeks muszą spełniać następujące wymagania:

- Wszystkie źródła mają następniki.
- Wszystkie połączenia zewnętrzne są podpięte do portów[5].
- Poprzednikiem żadnej parceli nie może być źródło.

Reguły reakcji mają reprezentować możliwe do wystąpienia zmiany w systemie. To, czy zmiana może nastąpić w danej sytuacji (reprezentowanej przez jakiś dwugraf B) jest sprawdzane poprzez wyszukanie redeksu R w dwugrafie B.

Dwugraf wejściowy $B : \langle 0, \emptyset \rangle \to \langle k, Y \rangle$ można zrekonstruować używając dowolnego wystąpienia redeksu, dwugrafu $C : \langle n, X \rangle \to \langle k, Y \rangle$ (nazywanego kon-

[5]W definicjach w języku angielskim ten wymóg jest sformułowany jako *no outer names are idle* gdzie *idle* oznacza niepodpięcie ani do portu, ani do połączenia wewnętrznego. Jednak ze względu na to, że zarówno redeks jak i reaktum nie mogą mieć połączeń wewnętrznych zostało to ograniczenie skrócone do samych portów.

tekstem reguły reakcji) oraz parametru $P : \langle 0, \emptyset \rangle \to \langle m, Y \rangle$

$$B = C \circ (R \otimes id_Y) \circ P$$

W powyższym równaniu dwugraf R jest równoważny z dokładnością do podstawy redeksowi reguły reakcji r. Parametr P jest dwugrafem o następujących cechach:

- Jego graf połączeń ma tylko połączenia zewnętrzne.
- Wszystkie połączenia zewnętrzne są połączone z dokładnie jednym portem.

W literaturze anglojęzycznej dwugrafy o powyższych cechach określane są *dyskretnymi* (ang. *discrete*). W kontekście reguł reakcji są one ponadto określane *parametrami* reguły.

Odzwierciedlenie zmiany w systemie wynikające z czynności reprezentowanej przez regułę reakcji r polega na podmianie redeksu na reaktum w powyższym równaniu, co można zapisać jako:

$$B' = C \circ (R' \otimes id_Y) \circ P$$

W powyższym równaniu dwugraf R' jest równoważny z dokładnością do podstawy reaktum reguły r.

Jako podsumowanie dotychczas omówionych kwestii związanych z dwugrafowymi systemami reakcyjnymi zostanie wyjaśnione znaczenie przymiotników określających typ reguły reakcji. Reguły reakcji omawiane w tej pracy są:

- *Śledzące* - ponieważ umożliwiają przypisanie wierzchołkom reaktum „odpowiadających im" wierzchołków redeksu, przy użyciu funkcji τ.
- *Parametryczne* - ponieważ umożliwiają zdefiniowanie tylko części sytuacji, w jakiej może dojść do zaistnienia czynności reprezentowanej przez regułę reakcji. Alternatywą do parametrycznych reguł reakcji są proste reguły reakcji (ang. *basic reaction rules*), w których zarówno redeks jak i reaktum są dwugrafami o interfejsach $\langle 0, \emptyset \rangle \to \langle n, X \rangle$.
- *Liniowe* - ponieważ nie umożliwiają zmiany parametrów reaktum względem tych z redeksu. Jest to szczególny przypadek reguł parametrycznych. W

ogólności parametry redeksu, których jest dokładnie tyle ile parceli, mogą mieć zmienianą kolejność w reaktum, a nawet mogą być kopiowane lub usuwane.

Dwa uziemione dwugrafy B_1, B_2 są w *relacji reakcji* r_1 zapisywanej jako $B_1 \xrightarrow{r_1} B_2$, jeżeli dla dowolnych dwóch dwugrafów $B \cong B_1$, $B' \cong B_2$ równania:

$$B = C \circ (R \otimes id_Y) \circ P \quad B' = C \circ (R' \otimes id_Y) \circ P$$

są spełnione dla jakiegoś dwugrafu C, dwugrafu dyskretnego P, a R i R' to redeks i reaktum reguły reakcji $r_1 = (R, R', \tau)$. Relacja reakcji r na ustalonym wcześniej zbiorze uziemionych dwugrafów będzie oznaczana symbolem $\xrightarrow{r}$. Relację tę można interpretować jako podzbiór zbioru $\mathcal{B} \times \mathcal{B}$, którego elementy oznaczają, że para dwugrafów w wchodząca w ich skład spełnia powyższe równania.

Śledzący dwugrafowy system reakcyjny[6] (ang. *tracking bigraphical reactive system* - TBRS) to para $(\mathcal{B}, \mathcal{R})$, która składa się ze zbioru uziemionych dwugrafów $\mathcal{B}$ oraz zbioru śledzących reguł reakcji $\mathcal{R}$.

Na zbiorze dwugrafów $\mathcal{B}$ można określić relację możliwych reakcji:

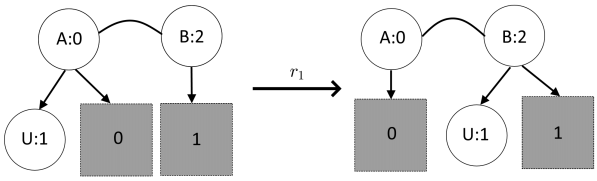
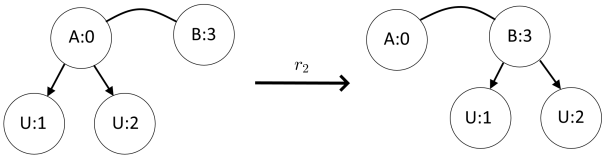
$$\xrightarrow{\mathcal{R}} = \bigcup_{r \in \mathcal{R}} \xrightarrow{r}$$

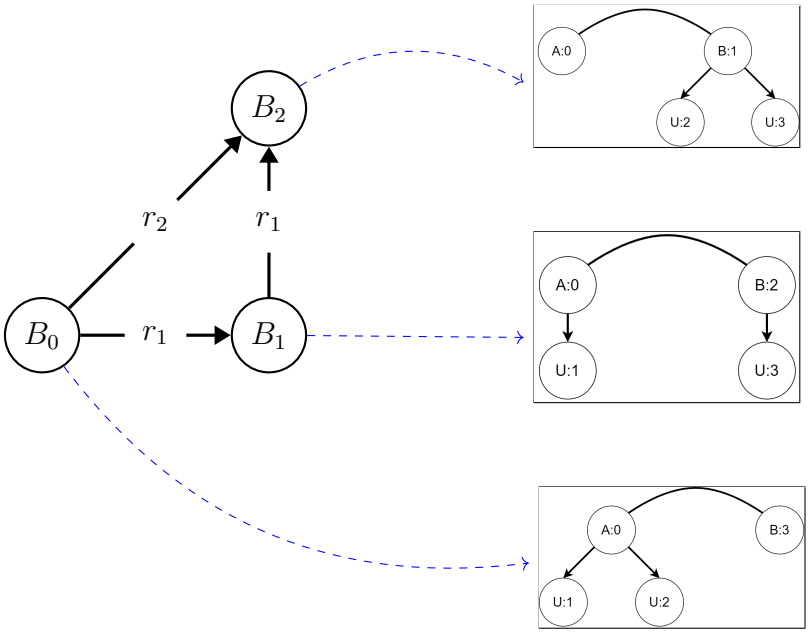
Można ją interpretować jako zbiór możliwych zmian, jakie mogą zajść pomiędzy stanami systemu.

Elementy przykładowego TBRS przedstawiono w tabeli 2.1 oraz na rysunku 2.7. Reguły reakcji w tym przykładzie można interpretować jako czynności umożliwiające przemieszczenie się dronów (oznaczonych sterownikami U) z obszaru A do obszaru B na dwa sposoby. Pierwszy sposób, utożsamiany z regułą r_1, polega na niezależnym przemieszczeniu jednego drona. Sposób w jaki została przedstawiona ta czynność oznacza, że jest ona wykonalna bez względu na to, czy przemieszczający się dron jest sam na aktualnym obszarze (parcela w wierzchołku typu A) oraz, czy na obszarze docelowym znajduje się jakiś dron (parcela

[6]Jest to definicja zmieniona względem tych przedstawionych w [62, 54] ze względu na to, że w tej pracy nie jest uwzględniana perspektywa teorii kategorii. Różnica pomiędzy definicjami we wspomnianych pracach, a tą tutaj polega na tym, że relacja reakcji na dwugrafach traktuje je jako dwugrafy abstrakcyjne, podczas gdy cały system reakcyjny jest konkretny.

Tabela 2.1: Elementy zbioru $\mathcal{R}$ dla śledzącego dwugrafowego systemu reakcyjnego. Funkcje τ dla obydwu są tożsamościami.

Nazwa reguły reakcji	Reprezentacja graficzna
r_1	
r_2	



Rysunek 2.7: Elementy zbioru $\mathcal{B}$ dla przykładowego śledzącego dwugrafowego systemu reakcyjnego wraz z relacją możliwych reakcji pomiędzy jego elementami. Relacja reakcji r_x wskazuje, która reguła reakcji z tabeli 2.1 może zostać zastosowana oraz jaki będzie wynik tej reakcji.

w wierzchołku typu B). Druga czynność, reprezentowana przez regułę r_2 polega na przemieszczeniu obydwu dronów jednocześnie tylko wtedy, gdy obydwa znajdują się na obszarze A, a na obszarze B nie ma nikogo. Na tym etapie nie jest to brane pod uwagę, ale przyjęto, że czynność utożsamiana z regułą reakcji r_1 będzie trwała dwa razy dłużej niż czynność utożsamiana z regułą reakcji r_2.

2.5 Systemy przejść

Dwugrafowe systemy reakcyjne wraz z relacją możliwych reakcji pozwalają określić, kiedy poszczególne czynności mogą zostać wykonane oraz jaki stan będzie wynikiem ich wykonania. W przypadku modelowania systemu, dla którego części elementów na późniejszym etapie projektowania opracowywany będzie sposób zachowania, taki poziom szczegółowości rozpatrywania dynamiki jest niewystarczający. Innymi słowy, nie wystarczy wiedzieć, jakie czynności mogą doprowadzić do pożądanej zmiany w systemie. Aby móc analizować dynamikę systemu w sposób umożliwiający określenie, jak poszczególne jego elementy mają działać aby zrealizować postawione zadanie należy dla każdej reakcji odpowiedzieć na następujące pytania:

- Które elementy systemu biorą udział w działaniu, którego wynikiem jest określony stan systemu?
- Które elementy stanu będącego wynikiem wspomnianej czynności są „pozostałością" stanu poprzedniego (przed wykonaną czynnością)? Należy tu również uwzględnić elementy, które zostały usunięte oraz te które powstały w jej wyniku.
- Jak wykonanie określonej czynności przez pewien podzbiór elementów wpływa na możliwość wejścia w ponowną interakcję z resztą elementów systemu?

Algorytmizacja udzielenia odpowiedzi na powyższe pytania będzie przedmiotem tego i następnych rozdziałów.

Formalizmem umożliwiającym udzielenie odpowiedzi na pierwsze dwa z powyższych pytań jest *etykietowany system przejść* [48] (ang. *labelled transition system*), dalej określany systemem przejść. W tej pracy wykorzystywane będą

śledzące systemy przejść (TTS) zaproponowane w [29].
Śledzący system przejść to siódemka: $TTS = (Agt, Red, Lab, Apl, Par, Res, Tra)$, w której kolejnymi elementami są:

- $Agt \subset \mathcal{B}$ - zbiór uziemionych dwugrafów;
- Red - zbiór redeksów reguł reakcji użytych do konstrukcji TBRS, na podstawie którego konstruowany jest TTS;
- Lab - zbiór etykiet reguł reakcji. Służy do rozróżnienia typów czynności (reguł reakcji), jakie mogą zostać wykonane w systemie;
- $Apl \subset Agt \times Lab$ - relacja wykonalności. Wskazuje kiedy (z których dwugrafów) mogą zostać wykonane poszczególne czynności (etykiety);
- $Par \subseteq \{V_r \to V_b \mid r \in Red,\ b \in Agt\}$ - zbiór funkcji udziału w przejściu. Każda z nich wskazuje, które elementy (wierzchołki) stanu wejściowego (dwugrafu z Agt) odpowiadają którym „rolom" w czynności (redeksie z Red). Będzie to zawsze iniekcja;
- $Res \subseteq \{V_{b_1} \to V_{b_2} \mid b_1, b_2 \in Agt\}$ - zbiór funkcji pozostałości. Każda z nich przypisuje elementom stanu będącego wynikiem działania, elementy stanu, z którego je wykonano;
- $Tra \subseteq Apl \times Agt \times Par \times Res$ - relacja przejścia. W dalszej części pracy elementy tego zbioru będą określane „przejściami" śledzącego systemu przejść. Rozpatrywane będą tylko przejścia, których funkcje uczestnictwa dla pary dwugrafów reprezentującej stan wejściowy i wynikowy różnią się dziedziną lub przeciwdziedziną.

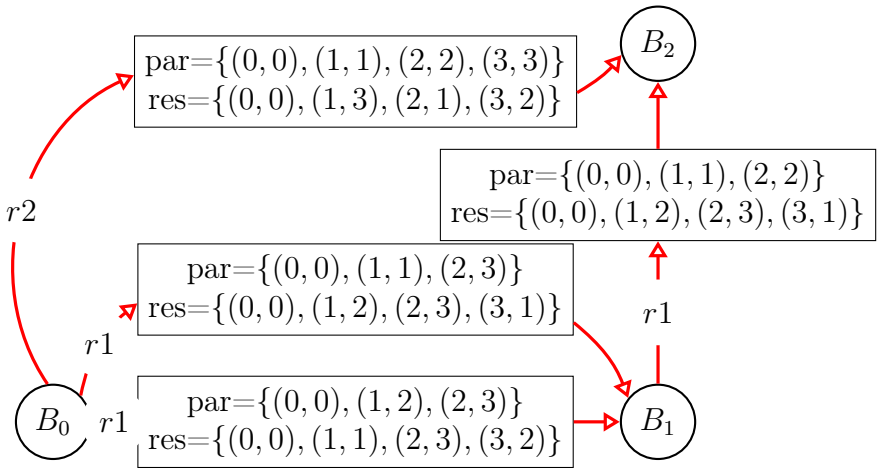
Śledzący system przejść jest konstruowany na podstawie wcześniej opracowanego śledzącego dwugrafowego systemu reakcyjnego. Sposób przekształcenia TBRS do TTS został opisany w załączniku B.

Kontynuując przykład z poprzedniego rozdziału, TTS będący wynikiem przekształcenia TBRS z rysunku 2.7 został formalnie zdefiniowany poniżej.

- $Agt = \mathcal{B}$ - zbiór dwugrafów uziemionych, w tym przypadku obydwa zbiory są identyczne;

Tabela 2.2: Śledzący system przejść przedstawiający elementy przestrzeni stanów. Jest on wynikiem przekształcenia TBRS z rysunku 2.7. Kolejne wiersze oznaczają pojedyncze przejścia w TTS.

Apl	Agt	Par	Res
$\langle B_0, r1\rangle$	B_1	$\{(0,0),(1,1),(2,3)\}$	$\{(0,0),(1,2),(2,3),(3,1)\}$
$\langle B_0, r1\rangle$	B_1	$\{(0,0),(1,2),(2,3)\}$	$\{(0,0),(1,1),(2,3),(3,2)\}$
$\langle B_0, r2\rangle$	B_2	$\{(0,0),(1,1),(2,2),(3,3)\}$	$\{(0,0),(1,3),(2,1),(3,2)\}$
$\langle B_1, r1\rangle$	B_2	$\{(0,0),(1,1),(2,2)\}$	$\{(0,0),(1,2),(2,3),(3,1)\}$



Rysunek 2.8: Graficzna reprezentacja przejść TTS zdefiniowanego w tabeli 2.2.

- *Red* - zbiór redeksów reguł r_1 i r_2 przedstawionych w tabeli 2.1;
- $Lab = \{r1, r2\}$ - zbiór etykiet do rozróżniania „typów" przejść tego TTS;
- zbiór *Tra* oraz wchodzące w jego skład elementy definiujące zbiory *Par* i *Res* zostały zdefiniowane w tabeli 2.2.

Graficzna reprezentacja powyższego TTS została zamieszczona na rysunku 2.8.

Dalej przedstawiono interpretację śledzącego systemu przejść z rysunku 2.8. Tak jak źródłowy dla niego TBRS składa się on z trzech stanów (dwugrafów), których zmiany są wynikiem jednej z dwóch czynności utożsamianych z regułami reakcji r_1 i r_2. W ramach tego TTS są trzy przejścia, oznaczające możliwe „konfiguracje" powyższych działań. Przykładowo ze stanu B_0 do B_1 można „przejść" wykonując działanie *r1*[7], w którym albo wierzchołek o identyfikatorze *1* pełni rolę utożsamianą z wierzchołkiem redeksu reguły r_1 o identyfikatorze *1* albo tę rolę pełni wierzchołek B_0 o identyfikatorze *2*. Różnice te są przedstawione w po-

[7]Różnica w zapisie r_1 i *r1* ma na celu rozróżnienie czynności reprezentowanej przez regułę reakcji r_1 od przejścia typu *r1*, które odzwierciedla pewną konfigurację jej wykonania.

staci funkcji *par*, będącej elementem każdego z przejść. W zależności od tego, kto uczestniczy w czynności, różne będą funkcje pozostałości (elementy zbioru *Res*), stanowiące również składnik przejść. Dla kontrastu można wskazać tylko jedno przejście ze stanu B_0 do B_2 pomimo, że są dwie możliwe kombinacje elementów stanu B_0 do ról redeksu reguły r_2. Jest to uproszczenie wspomniane w opisie zbioru *Tra*, w którym brane będą pod uwagę tylko te przejścia pomiędzy parą stanów, których funkcje uczestnictwa różnią się dziedziną lub przeciwdziedziną. Dla omawianego teraz przykładu wybrana funkcja uczestnictwa ma postać $par = \{(0,0),(1,1),(2,2),(3,3)\}$, a drugą (nie braną pod uwagę) kombinacją byłaby $par = \{(0,0),(1,2),(2,1),(3,3)\}$. Nie różnią się one zawartością ani dziedziny, ani przeciwdziedziny, a jedynie sposobem mapowania elementów. W związku z czym uwzględniane jest przejście tylko z jedną z nich.

Z pytań wymienionych wcześniej dotyczących możliwych reakcji w systemie, śledzące systemy przejść udzielają odpowiedzi na pierwsze dwa. Trzecie pytanie wymaga uwzględnienia czasu potrzebnego na wykonanie czynności utożsamianych z regułami reakcji i odpowiedź na nie będzie przedmiotem następnego rozdziału.

Rozdział 3

Metodyka projektowania zachowania roju dronów z użyciem dwugrafów

3.1 Przyjęte założenia względem projektowanych systemów

Przed wprowadzeniem formalnych definicji zdefiniowane zostaną główne założenia, ograniczenia oraz pojęcia związane z opracowaną metodyką.

W następnych rozdziałach zostanie zdefiniowana i omówiona metodyka umożliwiająca zaprojektowanie sposobu działania roju bezzałogowych pojazdów latających, który ma wykonać postawione zadanie. Metodyka tak jak wcześniej będzie rozumiana jako algorytm wraz ze sposobem przekształcenia elementów świata rzeczywistego na jego struktury. W dalszej części pracy proces projektowania zachowania elementów roju BPL będzie określany projektowaniem roju. Projektowany rój wraz z otoczeniem, w którym się znajduje można rozpatrywać jako system wieloagentowy (MAS), w którym drony są agentami. Używając tej analogii opracowana metodyka sprawdza się najlepiej dla systemów wieloagentowych, w których agenci są do siebie podobni, a także są w stanie realizować skomplikowane polecenia, działając w oparciu o zbiór działań elementarnych. Innymi słowy, systemy te mają cechy charakterystyczne rojów. Pierwotna wersja

zdefiniowanej tutaj metodyki została opisana w [29], a następnie rozwinięta w [28].

Za cel projektowania roju przyjmiemy opracowanie modelu dla każdego z zadań przed nim stawianych oraz konstrukcję harmonogramu działań, który pozwoli mu wykonać wszystkie postawione zadania. W tej pracy przyjęto, że rezultatem modelowania zadania, które ma być wykonane przez projektowany rój będzie następujący zbiór elementów:

- definicja zadania,
- dwugrafowy śledzący system reakcyjny,
- system przejść,
- przestrzeń stanów.

Drugi i trzeci składnik modelu zostały omówione w poprzednim rozdziale. Pozostałe będą szczegółowo zdefiniowane w dalszej części pracy.

Przed określeniem założeń względem modelowanych systemów zdefiniowana zostanie związana z nimi terminologia. Głównymi pojęciami używanymi w dalszej części pracy związanymi z opracowaną metodyką projektowania roju są:

- Zadanie - zbiór elementów świata rzeczywistego wraz z czynnościami w jakie mogą się angażować. Ponadto zadanie musi określać stan początkowy oraz stan(y) końcowy (końcowe). Przykładem zadania może być:
  „Na mapie w postaci kraty o wymiarach 3 na 3 znajdują się 2 drony, umieszczone na przeciwko siebie, po skosie. Każdy z dronów może przemieszczać się po mapie w pionie i poziomie oraz nawiązać połączenie z drugim, jeżeli obydwa znajdują się na tym samym obszarze. Celem zadania jest to, aby drony nawiązały ze sobą połączenie.”.

- Elementy zadania - element świata rzeczywistego, który ma znaczenie dla wykonywanego zadania lub abstrakcyjna struktura potrzebna do wykonania modelu. Elementami zadania mogą być zarówno ludzie, drony, obszary, nadajniki telekomunikacyjne itp. jak i np. dane będące w posiadaniu wcześniej wymienionych.

- Otoczenie realizowanego zadania - elementy zadania, dla których nie istotny jest upływ czasu w trakcie wykonywania zadnia. Formalna definicja istotności wpływu czasu jest trudna do podania. Przykładem obiektów, na które czas ma wpływ będą np. przemieszczające się drony lub obiekty, mające różnych właścicieli. Elementy zadania, na które czas nie miałby wpływu to np. obszary, pomiędzy którymi drony lub ludzie mogą się przemieszczać. Elementy otoczenia nie mogą inicjować zmian w systemie.

- Obiekty bierne - Elementy zadania, na które upływ czasu ma znaczenie, nie mogące inicjować zmian w systemie. Przykładem mogą być przedmioty przenoszone przez drony.

- Obiekty czynne (agenci) - Elementy zadania, które mogą inicjować zmiany w systemie, dla których upływ czasu ma znaczenie. Przykładem mogą być drony wykonujące zadanie.

- Misja - Sposób realizacji zadania.

Mając zdefiniowane podstawowe pojęcia można przejść do założeń oraz ograniczeń stawianych rozpatrywanym systemom. Poniżej przedstawiono założenia względem opracowywanego modelu w formie opisowej. Praktyczne implikacje tych założeń zostaną omówione w 3.2.
Opracowana metodyka projektowania roju dronów zakłada, że modelowany system spełnia następujące założenia:

1. Liczba obiektów czynnych i biernych jest stała w trakcie misji.

2. Zmiany w systemie mogą być wynikiem tylko działań agentów.

3. Dla każdego typu czynności, w którą jest zaangażowany określony zbiór elementów zadania jest tylko jeden możliwy rezultat.

4. Wszystkie elementy zadania mogą zakończyć swój udział w misji w dowolnym momencie.

5. Wszystkie obiekty uczestniczące w dowolnej czynności rozpoczynają ją i kończą jej wykonywanie w tym samym momencie. Oznacza to, że każdy obiekt uczestniczący we wspólnym działaniu jest zaangażowany równie długo.

Podsumowując powyższe można podkreślić następujące koncepcje:

- Projektowanie roju oznacza definiowanie sposobu, w jaki jego elementy mają się zachowywać.
- Projektowany rój ma wykonywać postawione zadania.
- Rój wraz z otoczeniem, w którym działa jest systemem (wieloagentowym).
- Modelowany jest system, w którym wykonywane jest pojedyncze zadanie stawiane przed rojem.
- Wynikiem projektowania jest model dla każdego zadania wraz harmonogramem działań, pozwalającym na realizację wszystkich zadań.

3.2 Przestrzeń stanów

Mając zdefiniowane pojęcia, ograniczenia oraz założenia względem modelowanych systemów można przystąpić do formalnego zdefiniowania etapów metodyki projektowania.

Przestrzeń stanów (SS) systemu to multigraf skierowany, w którym wierzchołki reprezentują nieizomorficzne[1] stany systemu, a łuki odzwierciedlają możliwe czynności zmieniające stan systemu. W przypadku kilku łuków reprezentujących taką samą czynność pomiędzy tą samą parą wierzchołków mamy do czynienia z różnymi elementami zadania, zaangażowanymi w taką samą czynność. Przykładowo w systemie składającym się z kilku osób oraz czajnika z zimną wodą jest kilka rozróżnialnych czynności, które doprowadzą ze stanu „zimna woda w czajniku" do „wrzątek w czajniku". Każda z tych czynności jest tego samego typu tj. „zagotowanie wody", ale będą się one różnić zaangażowanymi elementami systemu, czyli tym kto ją zagotował.

[1]Stany systemu w TBRS oraz TTS są reprezentowane przez dwugrafy nierównoważne względem siebie z dokładnością do zwięzłych podstaw. W przestrzeni stanów wszystkie stany zostają zredukowane do wierzchołków multigrafu skierowanego, co uniemożliwia ich analizę porównawczą. W związku z tym, stany w SS są w tej pracy określane nieizomorficznymi, co ma na celu podkreślić fakt, że nie można przedstawić żadnego z nich przy użyciu jakiegokolwiek innego stanu.

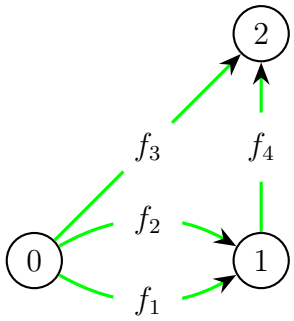
Przestrzenią n_s stanów systemu, w którym jest wykonywane zadanie i znajduje się n_o obiektów określimy ósemką:

$$SS = (S, E, M_E, L, I, C, T, M_f)$$

gdzie kolejnymi elementami są:

- $S = \{0, n_s - 1\}$ - zbiór stanów modelowanego systemu. Liczba stanów oraz same stany odpowiadają dwugrafom w źródłowym śledzącym systemie przejść.
- $E \subset \mathbb{N}$ - zbiór identyfikatorów łuków.
- $M_E \subset S \times E \times S$ - relacja trójczłonowa definiująca połączenia pomiędzy wierzchołkami.
- L - zbiór etykiet oznaczających możliwe typy zmian zachodzących w systemie. W celu określenia jakie zmiany w jakiej kolejności nastąpiły w systemie wprowadzimy ponadto zbiór $\mathbb{H} = \{l_t \mid l \in L, t \in \mathbb{N}\}$.
- $I \subset \mathbb{N}_1^2 \times \cdots \times \mathbb{N}_{n_o}^2$ - zbiór konfiguracji czasowo-strukturalnych (SaT). Celem tego zbioru jest przedstawienie upływu czasu dla obiektów w trakcie wykonywania misji. Realizowane jest to poprzez zawarcie informacji o kolejności obiektów zadania w reprezentacji dwugrafowej oraz momentu czasu, w którym każdy z nich się znajduje. Przykładowo dla konfiguracji $i_x = \langle(2,4),(1,3)\rangle, i_x \in I$ interpretacja jest następująca: obiekt o identyfikatorze *1* (para uporządkowana, w której pierwszy element ma wartość 1) znajduje się w momencie czasu 3. Z kolei obiekt o identyfikatorze *2* znajduje się w chwili czasu 4.

  Kolejność par uporządkowanych ma znaczenie przy wyznaczaniu polityki zachowania i zostanie zademonstrowana w rozdziale 3.3.
- $C \subset (I \times 2^{\mathbb{H}}) \cup \{0\}$ - zbiór przebiegów misji. Składa się z aktualnego stanu misji (konfiguracji czasowo strukturalnej) oraz historii czynności, jakie doprowadziły do aktualnego stanu. Element *0* oznacza nieosiągalny przebieg misji. Wymieniając zbiór możliwych kombinacji czynności $\{c_1, \ldots, c_x\}$ $c_i \in$



Rysunek 3.1: Przestrzeń stanów skonstruowana na podstawie śledzącego systemu przejść z tabeli 2.2. Definicje funkcji postępu misji zostały przedstawione w tabeli 3.1.

C, które doprowadziły do jakiegoś stanu używany będzie zapis: $c_1 + \cdots + c_x$. Wszystkie nieosiągalne przebiegi misji są nieistotne z punktu widzenia celu projektowania i zapis: $0 + c_1 + c_2 + 0 + c_3 + 0$ będzie równoważny: $c_1 + c_2 + c_3$.

- $T = \{f_i : C \times \mathbb{N} \to C \mid i \in \mathbb{N}\} \cup \{f_{null}\}$ - zbiór funkcji określających postęp misji. Każda z funkcji przyjmuje jako argument aktualny przebieg misji oraz liczbę oznaczającą, która z kolei czynność jest teraz rozpatrywana (pierwsza, druga, trzecia itd.). Zbiór $T_{i,j} \subset T$ zawiera wszystkie funkcje postępu misji umożliwiające przejście ze stanu i do stanu j.

- $M_f : E \to T$ - bijekcja przypisująca łukom funkcje postępu misji.

Śledzący system przejść można przekształcić w przestrzeń stanów. Sposób przekształcania kolejnych przejść w TTS został omówiony w załączniku C. W dalszej części tej pracy przyjęto, że wszystkie przejścia źródłowego TTS będą przekształcane w funkcje postępu misji.

Poniżej przedstawiono przykład zastosowania powyższych definicji. Śledzący dwugrafowy system reakcyjny (TBRS) z rozdziału 2.4 składający się z dwóch reguł reakcji został przekształcony w śledzący system przejść (TTS) zaprezentowany w tabeli 2.2. Dla takiego TTS odpowiadająca mu przestrzeń stanów została przedstawiona na rysunku 3.1. Wszystkie dwugrafy w TTS są przekształcane w stany w przestrzeni stanów. Liczba przejść w TTS definiuje liczbę funkcji postępu misji. Przyjmiemy, że czas wykonania czynności utożsamianej z regułą reakcji *r1* jest dwukrotnie dłuższy niż potrzebny na wykonanie czynności *r2*. Przekłada się to na definicje funkcji postępu misji, które zostały przedstawione w tabeli 3.1.

Tabela 3.1: Funkcje postępu misji dla przestrzeni stanów z rysunku 3.1. Przyjęto, że czynność reprezentowana przez etykietę *r1* zajmuje dwie jednostki czasu, podczas gdy czynność reprezentowana przez etykietę *r2* zajmuje jedną jednostkę czasu.

Funkcja	Definicja funkcji
$f_1(c,t)$	$\begin{cases} \left[\langle (b,y),(a,x+2)\rangle, \mathbb{H}' \cup \{r1^1_{t+1}\}\right] & c = [\langle (a,x),(b,y)\rangle, \mathbb{H}'] \\ 0 & c = 0 \end{cases}$
$f_2(c,t)$	$\begin{cases} \left[\langle (a,x),(b,y+2)\rangle, \mathbb{H}' \cup \{r1^2_{t+1}\}\right] & c = [\langle (a,x),(b,y)\rangle, \mathbb{H}'] \\ 0 & c = 0 \end{cases}$
$f_3(c,t)$	$\begin{cases} \left[\langle (a,z+1),(b,z+1)\rangle, \mathbb{H}' \cup \{r2^3_{t+1}\}\right] & c = [\langle (a,z),(b,z)\rangle, \mathbb{H}'] \\ 0 & c \neq [\langle (a,z),(b,z)\rangle, \mathbb{H}'] \end{cases}$
$f_4(c,t)$	$\begin{cases} \left[\langle (b,y),(a,x+2)\rangle, \mathbb{H}' \cup \{r1^4_{t+1}\}\right] & c = [\langle (a,x),(b,y)\rangle, \mathbb{H}'] \\ 0 & c = 0 \end{cases}$

Formalnie taka przestrzeń stanów będzie się składać z następujących elementów:

- $S = \{0,1,2\}$,
- $E = \{1,2,3,4\}$,
- $M_E = \{(0,1,1),(0,2,1),(0,3,2),(1,4,2)\}$,
- $L = \{r1^1, r1^2, r2^3, r1^4\} \quad \mathbb{H} = \{r1^1_t, r1^2_t, r2^3_t, r1^4_t \mid t \in \mathbb{N}\}$,
- $I \subseteq \mathbb{N}^2 \times \mathbb{N}^2$,
- $C \subset (I \times 2^{\mathbb{H}}) \cup \{0\}$,
- $T = \{f_1, f_2, f_3, f_4\}$,
- $M_f = \{(1,f_1),(2,f_2),(3,f_3),(4,f_4)\}$.

W dalszej części tego podrozdziału wymieniono konsekwencje przyjętych założeń, względem modelowanych systemów na przestrzeń stanów:

1. Stała liczba obiektów czynnych i biernych przekłada się na stałą liczbę par uporządkowanych, wchodzących w skład zbioru SaT.
2. Pewność rezultatu wykonywanych czynności skutkuje tym, że przestrzeń stanów jest multigrafem, a nie hipergrafem skierowanym. Gdyby działania wykonywane przez agentów mogły mieć różne rezultaty, to z każdego stanu (wierzchołka) jedna czynność (łuk) mogłaby skutkować różnymi stanami

wynikowymi (wierzchołkami). W praktyce oznacza to, że żadna czynność wykonywana w systemie nie może mieć kilku wyników, czyli np. nie może się nie udać.

3. Założenie, że wszystkie obiekty czynne mogą zakończyć misję w różnym momencie czasu jest uwzględnione w sposobie wyznaczania zachowania dla tych obiektów i będzie szczegółowo omówione w rozdziale 3.3.

4. Wymaganie rozpoczęcia udziału w czynności przez wszystkie obiekty w niej uczestniczące w tym samym momencie jest odzwierciedlone w definicjach funkcji postępu misji. Przykładowo funkcja f_3 w tabeli 3.1 odpowiada regule reakcji, w której obydwaj agenci (reprezentowani przez sterowniki typu U) przemieszczają się z obszaru typu A do obszaru typu B. Oznacza to, że obydwaj agenci wykonują czynność kooperacyjną. Muszą zatem rozpocząć i zakończyć ją w tym samym czasie. W związku z powyższym funkcja f_3 wymaga, aby elementy argumentu c oznaczające czas zaangażowanych obiektów miały identyczną wartość. Jeżeli tak nie jest, to aktualny przebieg misji zostaje przekształcony przez funkcję f_3 na nieosiągalny.
   Praktyczne implikacje takiej postaci funkcji postępu będą omówione w podrozdziale 3.3, poświęconym wyznaczaniu polityki zachowania dla agentów.

3.3 Polityka zachowania

Mając zdefiniowaną przestrzeń stanów projektowanego systemu oraz znając identyfikatory wierzchołków reprezentujących stany końcowe oraz identyfikator stanu początkowego można przystąpić do opracowania sposobu działania jego elementów.

Polityką zachowania będziemy określać marszrutę w przestrzeni stanów. Każda polityka zachowania musi spełniać niżej wymienione wymagania:

- Wdrożenie polityki zachowania musi wykonywać postawione zadanie. Innymi słowy, jej pierwszym wierzchołkiem musi być ten, który reprezentuje stan początkowy systemu, a ostatni musi oznaczać jego stan końcowy.

- Każda polityka zachowania musi być ciągła dla obiektów zadania. Należy

przez to rozumieć, że nie pozostawia ona luk czasowych w działaniu obiektu. Wymóg ten nie dotyczy elementów otoczenia.

Sposób wykonania zadania (misja) z użyciem konkretnej polityki zachowania będzie określany jako *scenariusz*.

Do zdefiniowania sposobu konstrukcji poprawnej polityki zachowania zostaną wprowadzone poniższe definicje. Przez *szereg* należy rozumieć skończoną sumę elementów.

- $K_s^t = c_1 + \cdots + c_m = \sum_{i=1}^{m} c_i \quad c_i \in C, s \in \{0, \cdots, n_s - 1\}, t \in \mathbb{N}$ - szereg, którego elementami są przebiegi misji składające się z t działań doprowadzające do stanu s. Funkcja $n_K : K_s^t \to \mathbb{N}$ zwraca liczbę elementów szeregu (liczbę m według oznaczeń).

- $F_{i,j}^t = \sum_{f \in T_{i,j}} f(x,t) \quad i,j \in \{0, \cdots, n_s - 1\}, x \in C$ - szereg funkcyjny, którego elementami są funkcje postępu misji ze stanu i do stanu j. Każda z tych funkcji przyjmuje jako argument zmienne x i t. Tylko wartość t jest określona w momencie konstruowania szeregu.

- $\mathbb{M}_K^t = \begin{bmatrix} K_0^t & \cdots & K_{n_s-1}^t \end{bmatrix}, t \in \mathbb{N}$ - macierz, której elementami są szeregi o długości t wskazujące na możliwe przebiegi misji prowadzące do kolejnych stanów systemu.

- $\mathbb{M}_F^t = \begin{bmatrix} F_{0,0}^t & \cdots & F_{0,(n_s-1)}^t \\ \cdots & \cdots & \cdots \\ F_{(n_s-1),0}^t & \cdots & F_{(n_s-1),(n_s-1)}^t \end{bmatrix}$ - macierz możliwych przejść pomiędzy stanami systemu.

Ponadto zdefiniowane zostaną dwie operacje:

- $K_s^t \circ F_{i,j}^t = \sum_{f \in F_{i,j}^t} \sum_{i=1}^{n_K(K_s^t)} f(c_i, t)$ - splot powyższych szeregów.

- $\mathbb{M}_K^{t+1} = \mathbb{M}_K^t \cdot \mathbb{M}_F^t$ - mnożenie macierzy typu $\mathbb{M}_K^t$ przez macierz typu $\mathbb{M}_F^t$. Elementy nowej macierzy są określone według wzoru:

$$K_j^{t+1} = \sum_{i=0}^{n_s-1} K_i^t \circ F_{i,j}^t$$

Analizując powyższe operacje, można wskazać następujące ich właściwości:

1. Splot szeregów K^t i F^t daje w wyniku szereg typu K^{t+1}. Zwiększenie wartości indeksu górnego wynika z założenia, że wszystkie elementy zbioru T (czyli funkcje postępu misji) różne od f_{null} rozszerzają zbiór $\mathbb{H}$, będący elementem przebiegu misji c.

2. Szereg $F_{i,j}$ zawiera funkcje postępu misji ze stanu i do stanu j, a szereg K_s zawiera znane przebiegi misji prowadzące do stanu s. W związku z tym, jedyne szeregi K, dla których splot $K_s \circ F_{i,j}$ ma sens, to takie, które prowadzą do stanu i.

3. Splot szeregów $K_i^t \circ F_{i,j}^t$ daje w wyniku szereg K_j^{t+1}, zawierający wszystkie znane przebiegi misji do stanu j.

4. Wiedząc, że macierz $\mathbb{M}_K^t$ jest jednowierszowa, mnożenie macierzy $\mathbb{M}_K^t \cdot \mathbb{M}_F^t$ daje w wyniku macierz $\mathbb{M}_K^{t+1}$, której elementy, zgodnie z powszechną definicją miałyby postać $K_j^{t+1} = \sum_{i=0}^{n_s-1} K_i^t \cdot F_{i,j}^t$. Zastępując niezdefiniowaną dla szeregów K i F operację mnożenia splotem otrzymujemy wzór na wartość kolejnych elementów macierzy $\mathbb{M}_K^{t+1}$, równy temu powyżej. Oznacza to, że pojedyncze mnożenie macierzy $\mathbb{M}_K^t \cdot \mathbb{M}_F^t$ daje w wyniku macierz, której kolejnymi elementami są szeregi, oznaczające wszystkie znane przebiegi misji, prowadzące do kolejnych stanów systemu. Liczebność zbioru $\mathbb{H}$ elementów tych szeregów jest o jeden większa od tych w macierzy $\mathbb{M}_K^t$.

W celu wyznaczenia wszystkich wariantów polityki zachowania o żądanej długości należy wykonać taką samą liczbę iloczynów $\mathbb{M}_K^t \cdot \mathbb{M}_F^t$, przy czym w każdym kolejnym iloczynie parametr t będzie ulegał zwiększeniu, a macierz $\mathbb{M}_K^t$ dla $t > 0$ będzie wynikiem poprzedniego iloczynu.

W celu zademonstrowania sposobu wykorzystania definicji wprowadzonych w tym rozdziale, powrócimy do przykładu przestrzeni stanów użytej w rozdziale 3.2. Pierwszym krokiem jest zdefiniowanie macierzy M_F^t oraz M_K^0. Macierz M_K^0 wyznacza początkowy stan systemu. Kolejne kolumny tej macierzy definiują, jakie konfiguracje czasowo-strukturalne może mieć modelowany system na początku swojego działania. W tym przypadku początkowy stany systemu będzie

utożsamiany z wierzchołkiem *0* w przestrzeni stanów z rysunku 3.1. Ponadto przyjmiemy, że agenci będący jednocześnie jedynymi obiektami w zadaniu rozpoczynają misję w tym samym momencie, oznaczonym umownie jako moment czasu 0. W związku z powyższym macierz M_K^0 będzie miała następującą postać:

$$\mathbb{M}_K^0 = \begin{bmatrix} [\langle (1,0),(2,0)\rangle, \emptyset] & 0 & 0 \end{bmatrix}$$

Następnym krokiem jest zdefiniowanie macierzy M_F^t. Przypomina ona macierz przejść, z tą różnicą, że zamiast liczby przejść pomiędzy wierzchołkami multigrafu skierowanego elementami tej macierzy są szeregi funkcyjne $F_{i,j}^t$. Dla omawianego przykładu macierz ta będzie miała postać:

$$\mathbb{M}_F^t = \begin{bmatrix} f_{null}(c,t) & f_1(c,t) + f_2(c,t) & f_3(c,t) \\ f_{null}(c,t) & f_{null}(c,t) & f_4(c,t) \\ f_{null}(c,t) & f_{null}(c,t) & f_{null}(c,t) \end{bmatrix} \equiv \begin{bmatrix} f_{null} & f_1 + f_2 & f_3 \\ f_{null} & f_{null} & f_4 \\ f_{null} & f_{null} & f_{null} \end{bmatrix}$$

Ostatnim etapem jest wymnażanie powyższych macierzy, zastępując $\mathbb{M}_K^0$ kolejnymi $\mathbb{M}_K^1 \cdots \mathbb{M}_K^n$, gdzie n oznacza albo oczekiwaną długość marszruty albo wartość, dla której jedna ze znalezionych marszrut jest satysfakcjonująca. Kolejne macierze $\mathbb{M}_K^n$ zostały przedstawione poniżej.

$$\mathbb{M}_K^1 = \mathbb{M}_K^0 \cdot \mathbb{M}_F^0 = \begin{bmatrix} [\langle (1,0),(2,0)\rangle, \emptyset] & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_{null}(c,0) & f_1(c,0) + f_2(c,0) & f_3(c,0) \\ f_{null}(c,0) & f_{null}(c,0) & f_4(c,0) \\ f_{null}(c,0) & f_{null}(c,0) & f_{null}(c,0) \end{bmatrix}$$

$$\mathbb{M}_K^1 = \begin{bmatrix} 0 & [\langle (2,0),(1,2)\rangle, \{r1_1^1\}] + [\langle (1,0),(2,2)\rangle, \{r1_1^2\}] & [\langle (1,1),(2,1)\rangle, \{r2_1^3\}] \end{bmatrix}$$

Interpretacja macierzy $\mathbb{M}_K^1$ jest następująca. Zawiera ona wszystkie polityki zachowania o długości 1 (indeks górny), którymi w tym przypadku będą dwie marszruty do stanu o identyfikatorze *1* (pierwsza kolumna licząc od zera) oraz jedna marszruta do stanu *2*. Wiedząc, że stan początkowy jest utożsamiany z identyfikatorem 0 wspomniane dwie marszruty mają postać: $0 \xrightarrow{f_1} 1$ i $0 \xrightarrow{f_2} 1$. Funkcje postępu misji są jednoznacznie rozróżnialne na podstawie etykiet $r1^1$ i $r1^2$ należących do zbioru L w przestrzeni stanów. Marszruta o długości 1 umożli-

wiająca przejście do stanu *2* ma postać $0 \xrightarrow{f_3} 2$. Konfiguracje SaT będące elementami macierzy informują, który obiekt i w jakim momencie będzie po wykonaniu wszystkich czynności, zaplanowanych w ramach polityki zachowania (pojedynczej marszruty). W przypadku marszruty $0 \xrightarrow{f_1} 1$ obiekty zadania będą miały konfigurację czasową $\langle(2,0),(1,2)\rangle$ co oznacza, że obiekt o identyfikatorze *1* będzie w chwili czasu 2, a obiekt o identyfikatorze *2* będzie nadal w chwili 0. Zmiana kolejności obiektów pełni funkcję techniczną i nie ma znaczenia w interpretacji na tym poziomie.

$$\mathbb{M}_K^2 = \mathbb{M}_K^1 \cdot \mathbb{M}_F^1 = \mathbb{M}_K^1 \cdot \begin{bmatrix} f_{null}(c,1) & f_1(c,1) + f_2(c,1) & f_3(c,1) \\ f_{null}(c,1) & f_{null}(c,1) & f_4(c,1) \\ f_{null}(c,1) & f_{null}(c,1) & f_{null}(c,1) \end{bmatrix}$$

$$\mathbb{M}_K^2 = \Big[0 \quad 0 \quad [\langle(1,2),(2,2)\rangle, \{r1_1^1, r1_2^4\}] + [\langle(2,2),(1,2)\rangle, \{r1_1^2, r1_2^4\}]\Big]$$

Sposób interpretacji macierzy $\mathbb{M}_K^2$ jest identyczny jak poprzednio. Zawiera ona wszystkie warianty polityki o długości 2. Istnieją tylko dwie takie marszruty i prowadzą do stanu 2 (druga kolumna licząc od 0).

3.4 Harmonogram działań

Jednym z celów projektowania roju jest opracowanie harmonogramu działań dla elementów zadania. Należy tutaj przypomnieć, że przez elementy zadania rozumiemy obiekty świata rzeczywistego, które są istotne w trakcie jego wykonywania. Mając zbiór elementów, które biorą udział w wykonywaniu zadania, harmonogram ma na celu udzielenie odpowiedzi na pytania:

- W jakich czynnościach poszczególne elementy zadania biorą udział?
- Jaka jest rola wskazanego elementu zadania w wykonywanej czynności?
- Kiedy wskazane czynności są rozpoczynane oraz ile trwa ich wykonywanie.

Z wymagań stawianych polityce zachowania oraz modelowanym systemom wynikają następujące cechy, jakie posiada każdy harmonogram działań:

1. Elementy zadania nie mogą dołączyć do trwającej czynności ani skończyć jej przed innymi elementami (wszyscy uczestnicy czynności zaczynają i kończą razem).

2. Elementy zadania nie mogą zmieniać swojej roli w trakcie czynności, którą wykonują (każda czynność kończy się w przewidywany sposób).

3. Wykonanie wszystkich czynności tego samego rodzaju trwa tyle samo czasu.

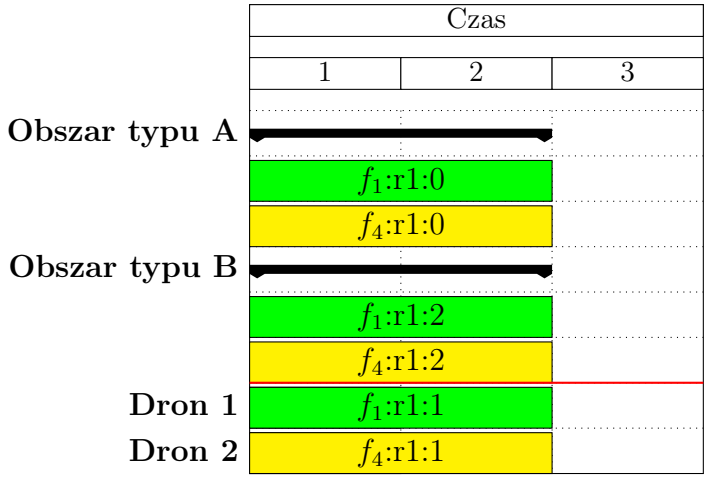
Wszystkie elementy zadania w trakcie jego wykonywania będą rozróżnialne przy użyciu identyfikatorów ze zbioru, który będzie w dalszej części pracy oznaczany symbolem $\mathcal{E}$. W rozdziale poświęconym metodzie weryfikacji modeli opracowanych przy użyciu omawianej tutaj metodyki tym samym symbolem określany będzie zbiór unikalnych identyfikatorów (UI).
Formalnie *harmonogram działań* będziemy definiować jako trójkę:

$$Schedule = (Assignment, role, duration)$$

składającą się z:

- Relacji binarnej $Assignment : \mathcal{E} \times (T \times Lab \times \mathbb{N})$, która przypisuje elementom zadania funkcję postępu misji, etykietę czynności, którą ta funkcja reprezentuje oraz moment rozpoczęcia wykonywanego działania.
  Nie może być to funkcja, gdyż elementy zadania mogą brać udział w różnych czynnościach ($\mathcal{E} \rightarrow T \times Lab \times \mathbb{N}$). Ponadto w tym samym czasie te same elementy otoczenia mogą brać udział w różnych czynnościach ($\mathcal{E} \times \mathbb{N} \rightarrow T \times Lab$).
  Zbiór T został omówiony w ramach definicji przestrzeni stanów i oznacza zbiór funkcji postępu misji. Zbiór Lab został wprowadzony w ramach definicji śledzącego systemu przejść i oznacza zbiór etykiet reguł reakcji. Zbiór liczb naturalnych $\mathbb{N}$ oznacza kolejne momenty trwania misji.

- Funkcji $role : \mathcal{E} \times (T \times Lab \times \mathbb{N}) \rightarrow V_r$ przypisującej pełnioną rolę elementom zadania, biorącym udział we wskazanej czynności . Rola w czynności jest reprezentowana przez wierzchołek redeksu reguły reakcji.
  Zbiór V_r oznacza zbiór wierzchołków redeksu reguły reakcji r.

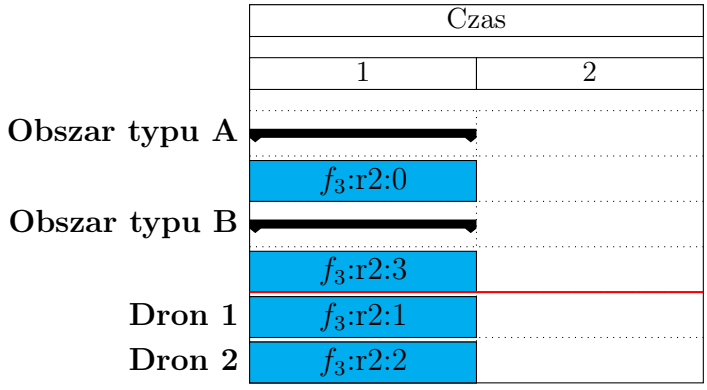


Rysunek 3.2: Harmonogram działań dla wszystkich elementów zadania z przykładu wprowadzającego, utworzony z marszruty $0 \xrightarrow{f_1} 1 \xrightarrow{f_4} 2$. Obszary typu A i B, będące elementami otoczenia biorą udział w kilku czynnościach jednocześnie. Drony (obiekty typu U) uczestniczą równolegle w różnych działaniach tego samego typu, angażujących te same obszary w ten sam sposób, a same pełnią identyczne role. Kolory mają na celu rozróżnić różne funkcje postępu misji (czynności). Identyfikatory elementów zadania zostały pominięte w tym harmonogramie.

- Funkcji *duration* : $T \to \mathbb{N}$, która każdej czynności (reprezentowanej przez funkcję postępu) przypisuje długość jej trwania.

  Zbiór liczb naturalnych $\mathbb{N}$ zawiera możliwe czasy trwania czynności reprezentowanych (pośrednio) przez funkcje postępu misji.

Z harmonogramów można odczytać w jakie czynności są angażowane poszczególne obiekty oraz jaką pełnią w nich role. Do zademonstrowania zostaną użyte dwa harmonogramy, utworzone na podstawie dwóch wariantów polityki zacho-



Rysunek 3.3: Harmonogram działań dla wszystkich elementów zadania z przykładu wprowadzającego, utworzony z marszruty $0 \xrightarrow{f_3} 2$.

wania, omówionych w poprzednim rozdziale. Pierwszy z nich zamieszczony na rysunku 3.2 przedstawia przebieg misji realizowanej na podstawie polityki zachowania $0 \xrightarrow{f_1} 1 \xrightarrow{f_4} 2$. Formalnie ten harmonogram będzie się składał z elementów:

- $Assignment = \left\{ \begin{array}{l} (1,(f_1,\mathrm{r1},0)),(1,(f_4,\mathrm{r1},0)), \\ (2,(f_1,\mathrm{r1},0)),(2,(f_4,\mathrm{r1},0)), \\ (3,(f_1,\mathrm{r1},0)),(4,(f_4,\mathrm{r1},0)) \end{array} \right\}$,
- $role = \left\{ \begin{array}{l} ((1,(f_1,\mathrm{r1},0)),0),((1,(f_4,\mathrm{r1},0)),0), \\ ((4,(f_1,\mathrm{r1},0)),2),((4,(f_4,\mathrm{r1},0)),2), \\ ((2,(f_1,\mathrm{r1},0)),1),((3,(f_4,\mathrm{r1},0)),1) \end{array} \right\}$,
- $duration = \{(f_1,2),(f_4,2)\}$.

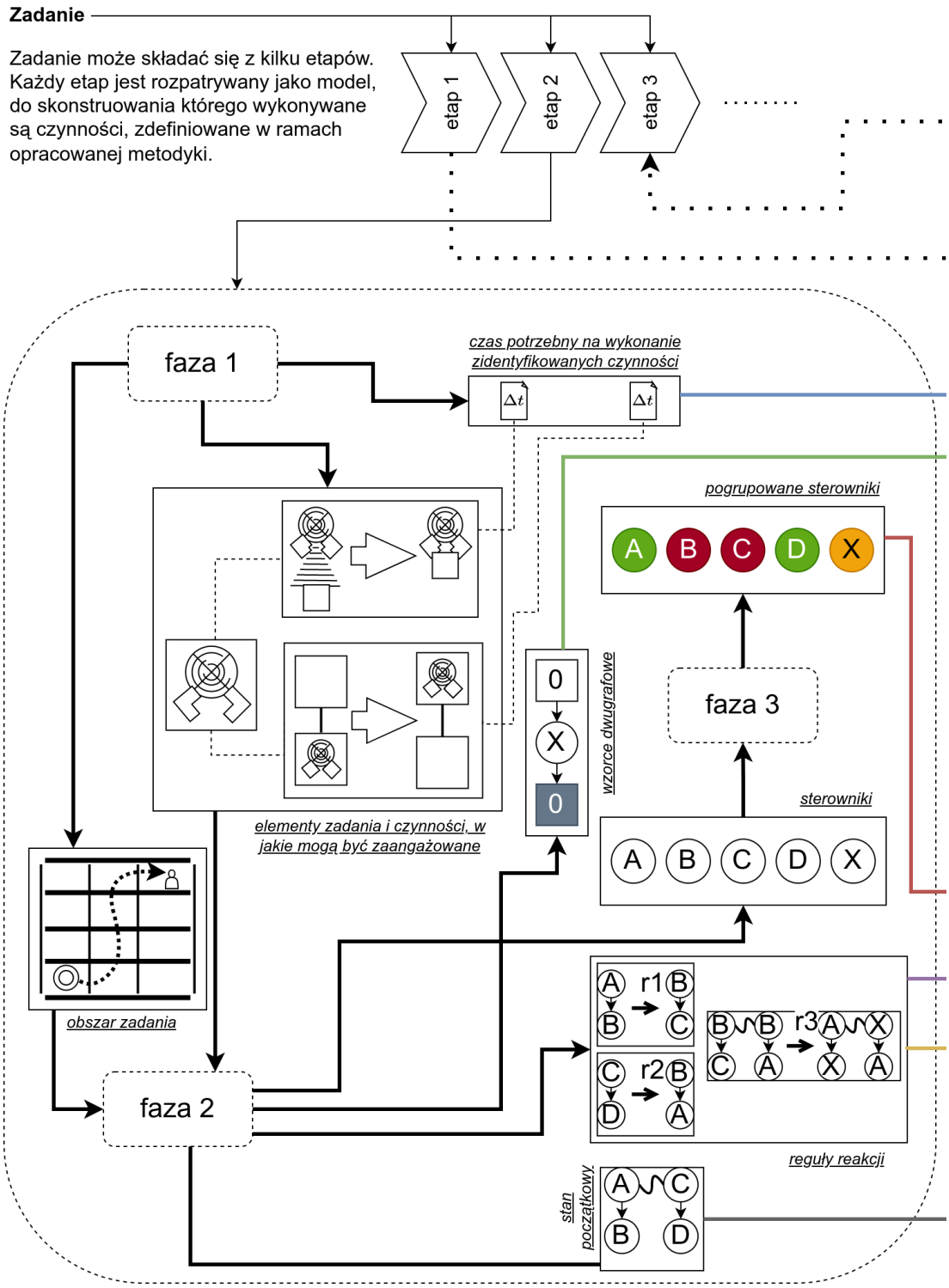
Warto zaznaczyć, że taki sam harmonogram[2] będzie wynikiem przekształcenia $0 \xrightarrow{f_2} 1 \xrightarrow{f_4} 2$. Drugi harmonogram, przedstawiony na rysunku 3.3, bazuje na polityce zachowania $0 \xrightarrow{f_3} 2$. Wykorzystując obydwa można wykonać to samo zadanie, ale sposób ich realizacji będzie inny w obydwu przypadkach.

W przypadku, gdy część działań wykonywanych w trakcie misji zajmuje nieproporcjonalnie mniej czasu niż pozostałe, można to przedstawić poprzez oznaczenie odpowiadających im funkcji postępu misji jako wymagających 0 jednostek czasu w funkcji *duration*. Działaniami tego typu mogą być np. wewnętrzne rekonfiguracje agentów.

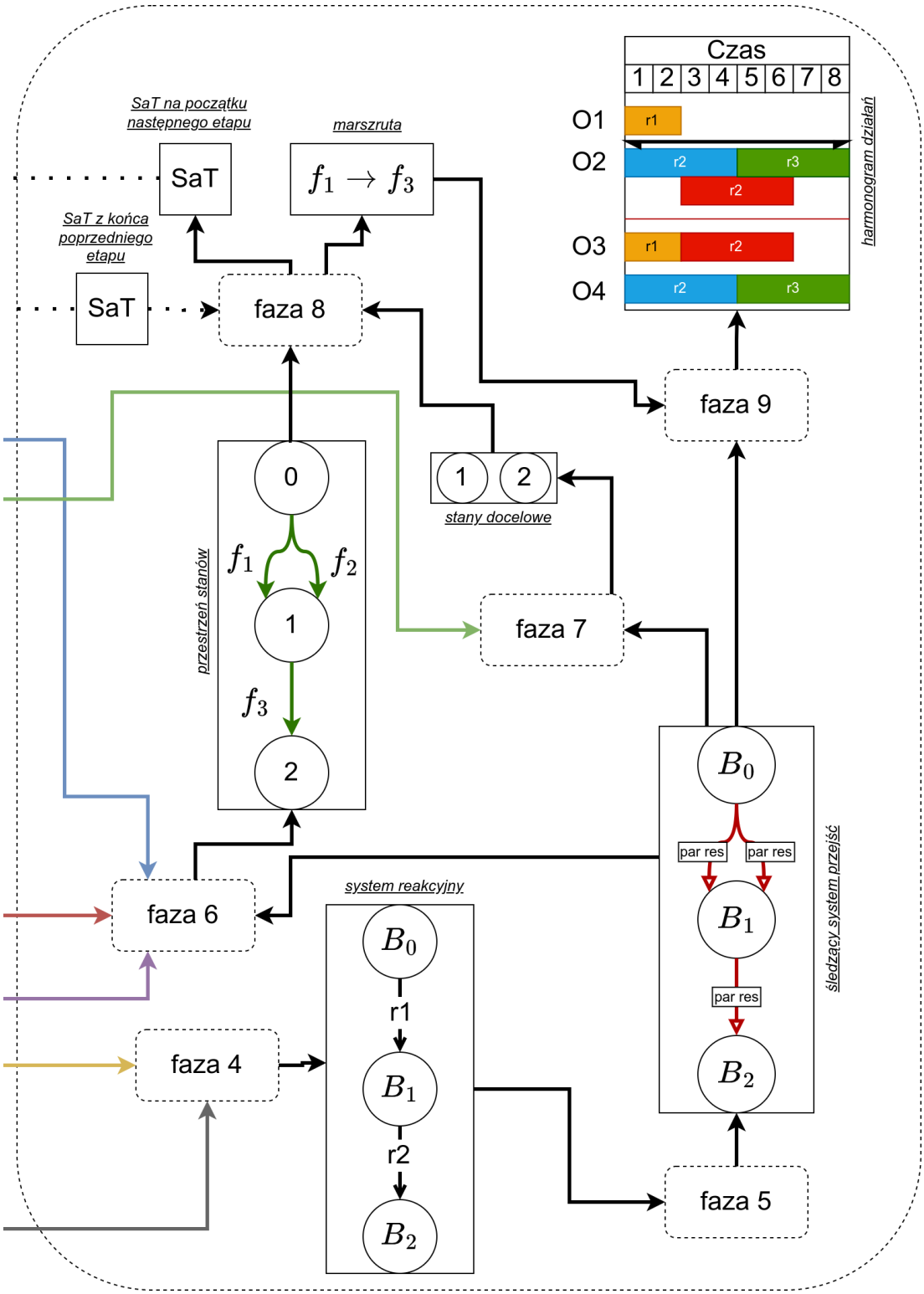
Sposób konstrukcji harmonogramu na podstawie polityki zachowania nie zostanie tutaj przedstawiony. W rozdziale 4.2 zdefiniowano oraz omówiono algorytm weryfikacji modeli MAS, opracowanych z użyciem omówionej metodyki. Jego rezultatem poza oceną poprawności jest również harmonogram w postaci zdefiniowanej w tym rozdziale.

[2] Z dokładnością do funkcji postępu misji.

3.5 Całościowy obraz metodyki



Rysunek 3.4: Całościowy obraz opracowanej metodyki - czynności wykonywane przez projektanta.



Rysunek 3.5: Całościowy obraz opracowanej metodyki - czynności wykonywane przez oprogramowanie.

Celem tego podrozdziału jest dokonanie całościowego podsumowania opracowanej metodyki. Przedstawiono tutaj, jak oraz kiedy wprowadzone struktury oraz algorytmy są wykorzystywane w trakcie całego procesu projektowania zachowania roju dronów. Dokonano również podziału na czynności w ramach metodyki, które muszą być wykonywane „ręcznie" przez projektanta oraz te, których wykonanie można zalgorytmizować i przekazać do wykonania komputerowi. Podział ten został wyznaczony w oparciu o oprogramowanie, które zostało przygotowane na potrzeby tej pracy. Oznacza to, że wszystkie czynności (określane jako *fazy* w dalszej części tego podrozdziału), które zostały zaliczone do miana algorytmizowalnych zostały zaimplementowane w ramach jednego z narzędzi opracowanych na potrzeby tej rozprawy [22, 24, 26, 23].

Postawione przed rojem zadanie można podzielić na etapy. Korzystając z opracowanej metodyki projektowania roju dronów, dla każdego etapu zadania należy wykonać 9 etapów projektowania, których ostatecznym wynikiem jest harmonogram działań oraz konfiguracja SaT, która może zostać wykorzystana w przypadku istnienia dalszych etapów zadania. Dla rozróżnienia etapów zadania od etapów projektowania, w dalszej części tego podrozdziału etapy projektowania będą określane *fazami*. Wspomniane fazy można krótko opisać w następujący sposób:

- Identyfikacja istotnych elementów świata rzeczywistego - faza 1.
- Wyrażenie zidentyfikowanych elementów przy użyciu języka dwugrafów - faza 2.
- Pogrupowanie zidentyfikowanych elementów na obiekty czynne, bierne lub elementy środowiska - faza 3.
- Konstrukcja TBRS - faza 4.
- Przekształcenie TBRS w TTS - faza 5.
- Przekształcenie TTS w SS - faza 6.
- Wyznaczenie stanów docelowych - faza 7.
- Wyznaczenie polityki zachowania w SS - faza 8.

- Przekształcenie polityki zachowania w harmonogram działań - faza 9.

Każda z wyżej wymienionych faz może być traktowana jako metoda, która wymaga określonych elementów jako argumentów, a której wynikiem są zdefiniowane wcześniej struktury.

Pierwsza faza polega na zidentyfikowaniu istotnych, z punktu widzenia rozpatrywanego etapu, elementów świata rzeczywistego. Takimi elementami będą zazwyczaj obiekty, ich stany, możliwości oraz czynności jakie mogą wykonywać, czas potrzebny na realizację tych czynności i obszar, na którym wykonywany będzie analizowany etap zadania.

Druga faza polega na „przetłumaczeniu" zidentyfikowanych w fazie pierwszej elementów na język dwugrafów. Wejściem tej fazy są zidentyfikowane elementy, a wynikiem:

- zbiór sterowników, które reprezentują obiekty świata rzeczywistego;
- stan początkowy w postaci dwugrafu;
- zbiór reguł reakcji;
- zbiór wzorców dwugrafowych, definiujących stan docelowy.

Trzecia faza polega na grupowaniu zdefiniowanych w poprzedniej fazie sterowników. Przyjmuje jako argument zbiór $\mathcal{K}$ sygnatury wszystkich wcześniej zdefiniowanych dwugrafów tj. wchodzących w skład reguł reakcji, wzorców dwugrafowych oraz dwugrafu reprezentującego stan początkowy etapu zadania. Wynikiem są trzy zbiory rozłączne, których suma zawiera wszystkie sterowniki podane na wejściu. W praktyce potrzebne jest tylko zidentyfikowanie, które sterowniki reprezentują obiekty czynne i bierne w zadaniu. Wszystkie dalej opisywane fazy mogą zostać zautomatyzowane.

Faza czwarta polega na konstrukcji TBRS. Wejściem jest stan początkowy oraz zbiór reguł reakcji. Wyjściem jest system reakcyjny ze zdefiniowaną nad nim relacją reakcji.

Faza piąta polega na konstrukcji TTS. Argumentem do tego potrzebnym jest wcześniej skonstruowany TBRS. Przygotowana biblioteka w języku OCaml [22] umożliwia bezpośrednią konstrukcję TTS, na podstawie stanu początkowego w

postaci dwugrafu oraz zbioru śledzących reguł reakcji. Przez bezpośredniość należy tutaj rozumieć pominięcie etapu konstrukcji TBRS, który następnie jest przekształcany w TTS.

Szósta faza polega na przekształceniu opracowanego w poprzednim kroku TTS w przestrzeń stanów. Argumentami wejściowymi są:

- śledzący system przejść;
- informacje o czasie potrzebnym na wykonanie poszczególnych czynności (reguł reakcji);
- grupa sterowników reprezentujących obiekty czynne i bierne w zadaniu.

Wynikiem fazy szóstej jest przestrzeń stanów. Jeden z programów wchodzących w skład pakietu programowego [26] umożliwia przekształcenie TTS w SS.

Celem fazy siódmej jest wyznaczenie wierzchołków w przestrzeni stanów, które odpowiadają dwugrafom reprezentującym akceptowalny stan docelowy. Argumentami wejściowymi tej fazy są TTS i wzorce dwugrafowe, definiujące stan docelowy. Wynikiem jest zbiór indeksów wierzchołków przestrzeni stanów. Podobnie jak poprzednio, program demonstrujący możliwość automatyzacji tej fazy wchodzi w skład pakietu [26].

Faza ósma ma na celu wyznaczenie polityki zachowania, realizującej rozpatrywany etap zadania. Przyjmuje ona jako argumenty:

- przestrzeń stanów;
- zbiór indeksów stanów, które reprezentują stany docelowe;
- (opcjonalnie) konfigurację SaT z końca poprzedniego etapu zadania.

Wynikiem fazy ósmej jest marszruta w przestrzeni stanów (polityka zachowania) oraz konfiguracja SaT na koniec tego etapu zadania, wykonanego przy użyciu opracowanej polityki. Do wyznaczania polityki zachowania przygotowano bibliotekę programową[3] [23] wraz z fasadą do jej wykorzystania.

[3]Przygotowano dwie wersje tego oprogramowania. Pierwsza (wcześniejsza) wersja została przygotowana w języku OCaml. Ze względu na brak możliwości konstrukcji wielowątkowych programów w aktualnej wersji tego języka, przygotowano drugą wersję w języku F#.

Ostatnia faza ma na celu przekształcenie wybranej polityki w harmonogram działań. Argumentami tej fazy (wykonywanej przy użyciu algorytmu przedstawionego i omówionego w następnym rozdziale) są przekształcana marszruta i TTS. Wynikiem fazy dziewiątej jest harmonogram działań dla elementów zadania, uczestniczących w wykonaniu rozpatrywanego etapu. Przykładowa implementacja procedury przekształcenia polityki zachowania w harmonogram działań została zamieszczona w repozytorium [24].

Wszystkie wyżej wymienione i opisane fazy wraz z artefaktami, na których operują, zostały przedstawione na rysunkach 3.4 i 3.5. Pierwszy z tych rysunków zawiera fazy, do wykonania których wymagane jest bezpośrednie zaangażowanie projektanta. Na drugim rysunku zamieszczono tylko fazy, które mogą zostać przekazane do wykonania specjalistycznemu oprogramowaniu.

Rozdział 4

Metoda weryfikacji modeli systemów wieloagentowych

4.1 Warunki poprawności modelu MAS

W tym rozdziale zostanie zdefiniowany oraz omówiony algorytm do weryfikacji modeli systemów wieloagentowych, opracowanych przy użyciu metodyki opisanej w rozdziale 3. Poza samą weryfikacją poprawności jego rezultatem będzie harmonogram działań dla elementów zadania, w rozumieniu definicji z rozdziału 3.4. Koncepcje tutaj przedstawione abstrahują od interpretacji elementów TBRS i TTS modelu opracowanego z użyciem metodyki opisanej w rozdziale 3. Z tego powodu w ramach tego rozdziału oraz odnosząc się do zagadnień w nim poruszanych modelowane systemy nie będą ograniczane do rojów dronów wraz z otoczeniem, w którym się znajdują, a zamiast tego rozpatrywane będą ogólnie jako systemy wieloagentowe.

Poprawność modelu MAS składającego się z: definicji zadania, TBRS, TTS oraz SS będziemy oceniać w odniesieniu do pojedynczej polityki zachowania. Model MAS będzie uznawany za poprawny, jeżeli dla wybranej polityki zachowania można zdefiniować stany systemu (w postaci dwugrafu) w kolejnych momentach czasu. Oznacza to między innymi, że wszystkie czynności reprezentowane przez funkcje postępu misji będące elementami polityki zachowania (marszruty w SS), które mają się odbywać w tym samym przedziale czasu muszą być od siebie niezależne. Nie jest to sprawdzane na etapie generowania marszrut, ponieważ

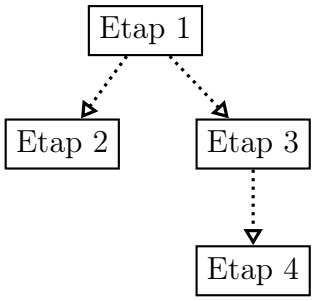
przestrzeń stanów abstrahuje od reprezentacji dwugrafowej stanów systemu. Tak zdefiniowany warunek poprawności modelu oznacza również potrzebę weryfikacji, czy poprawnie zdiagnozowano, które elementy zadania należą do grupy obiektów, a które do otoczenia. Błędna kategoryzacja skutkuje bowiem konstrukcją przestrzeni stanów, w której część elementów zadania może brać udział w czynnościach w dowolnym momencie oraz stanie systemu, gdy faktycznie mogły by one brać udział w czynnościach tylko w określonych sytuacjach.

Główną trudnością związaną z definiowaniem stanów systemu na podstawie polityki zachowania jest to, że jej elementy nie muszą odzwierciedlać kolejności czynności wykonywanych w systemie. Jeżeli zmiany w systemie reprezentowane przez funkcje postępu misji następują w innej kolejności niż elementy marszruty to zbiór przejść w TTS odpowiadający zmianom systemu w kolejnych przedziałach czasu będzie inny, niż to by wynikało z prostego przekształcenia tych funkcji z powrotem do przejść w TTS. Przykład zadania, w którym szczegółowo omówiono ten rodzaj trudności zamieszczono w rozdziale 5.2.

Z kolei analiza niezależności zachodzących w systemie zmian jest bezpośrednio powiązana z podziałem elementów zadania na: elementy otoczenia, obiekty bierne i aktywne. W przestrzeni stanów synchronizacja działań ma miejsce tylko dla obiektów (biernych i aktywnych). Oznacza to, że zaangażowanie elementów otoczenia w jakąś czynność nie może zmieniać, a tym bardziej uniemożliwiać możliwości zaangażowania ich w inne działania. Podobny problem omówiono w [62], gdzie rozpatrywano reguły reakcji, które mogą być wykonywane w dowolnej kolejności. We wskazanej publikacji reguły te były określane *kontekstowymi* (ang. *contextual reaction rules*). Zdefiniowano tam reguły reakcji, które można zapisać jako:

$$C \circ R \rightarrow C \circ R'$$

gdzie C oznacza *kontekst* reguły reakcji, a R i R' są parametrami odpowiednio redeksu i reaktum. Wynikiem tam przedstawionym był wniosek, że reguły reakcji, których tylko konteksty mają część wspólną, nie wykluczają się wzajemnie. Analogicznym rozwiązaniem problemu zdefiniowanego w tej pracy byłoby określenie największego wspólnego kontekstu reguł reakcji i przypisanie wszystkich jego elementów do grupy elementów otoczenia. Problem omawiany tutaj jest



Rysunek 4.1: Diagram zależności pomiędzy etapami weryfikacji modelu MAS.

bardziej skomplikowany, ponieważ w projektowanych systemach może pojawić się sytuacja[1], że fragment reguły reakcji, który byłby częścią R lub R' należałby do grupy elementów określanych jako elementy otoczenia.

Pomimo braku formalnych metod umożliwiających sprawdzenie, które sterowniki wierzchołków dwugrafu mogą być zakwalifikowane do grupy elementów otoczenia należy pamiętać, że dowolny element zadania, który może być w posiadaniu więcej niż jednego innego elementu, nie powinien być zaliczany do grupy elementów otoczenia.

4.2 Algorytm weryfikacji modelu

Niżej opisany algorytm weryfikacji modeli MAS składa się z 4 etapów. Etapy zostały ponumerowane w kolejności ich wykonywania. W dalszej części tego rozdziału etapy omawiane będą w kolejności takiej, aby nowo wprowadzane definicje formalne były widoczne w aktualnie omawianym etapie. Pomiędzy etapami istnieje zależność hierarchiczna, którą przedstawiono na rysunku 4.1. Poszczególne etapy poza podziałem proceduralnym (w rozumieniu podziału na funkcje) można podzielić ze względu na zadania, które mają zostać spełnione, a które mają pomóc w odpowiedzi na pytanie, czy weryfikowany model jest poprawny.

Inaczej niż w rozdziałach poświęconych metodyce projektowania MAS omawiając kolejne etapy weryfikacji nie będą załączone szczegółowe przykłady. Dla dwóch etapów (drugiego i czwartego) przedstawiono przykładowe dane wejściowe wraz z wynikiem algorytmu danego etapu. Szczegółowy przebieg działania al-

[1]Przykładem mogą być znaczniki obszarów, z których pobierane będą dane w przykładzie z rozdziału 5.4.2. Nie mogą być one traktowane jako obiekty, ponieważ pojawiają się w trakcie misji.

gorytmu weryfikacji modeli MAS został przedstawiony w rozdziałach 5.2 i 5.3. Załączniki D oraz F poświęcono szczegółowemu opisowi procedur, będących częścią etapu 2 i 4.

W dalszej części tego rozdziału istotnym będzie, aby odróżniać stan systemu od stanu scenariusza. Przez *stan systemu* należy rozumieć dowolny stan ze zbioru *Agt* w TTS. *Stanem scenariusza* będziemy określać dwugrafy utożsamiane ze stanem systemu w trakcie wykonywania zadania, w jakimś momencie czasu. Innymi słowy, mówiąc o stanie systemu potrzebujemy znać tylko reprezentujący go dwugraf. Odwołując się do stanu scenariusza potrzeba ponadto wiedzieć jaki scenariusz, czyli zadanie realizowane konkretną polityką zachowania jest realizowany oraz moment czasu.

Algorytm weryfikacji modeli MAS, którego poszczególne etapy będą definiowane w następnych rozdziałach można scharakteryzować następująco. Każdy wierzchołek dwugrafu, który ma odzwierciedlać stan scenariusza reprezentuje jakiś element zadania. Różne elementy mogą być tego samego typu, uczestniczyć w takich samych, a nawet tych samych czynnościach. Istnieje więc potrzeba ich jednoznacznego rozróżnienia w trakcie całej misji. Przez rozróżnienie należy rozumieć możliwość wskazania tych samych elementów zadania w różnych stanach scenariusza. Należy tutaj podkreślić, że funkcje pozostałości (*res*) w przejściach TTS nie są wystarczającym mechanizmem analizy czasowej scenariuszy. Jest tak z dwóch powodów. Pierwszym jest to, że funkcja pozostałości pozwala określić jedynie, które elementy stanu wynikowego są „pozostałością" po stanie wejściowym. Rozwiązaniem tego problemu mogłoby być „śledzenie" z użyciem zbioru przejść. Drugim problemem jest to, że elementy polityki zachowania nie są dodawane do marszruty w kolejności, w jakiej czynności przez nie reprezentowane będą wykonywane w systemie. Oznacza to, że stan scenariusza w określonym momencie będzie konstruowany w oparciu o poprzedni stan, a także zbiór przejść w TTS. Niekoniecznie będą one odpowiadały przejściom, z których powstały funkcje postępu misji, będące składnikami marszruty. W celu wybrania odpowiednich przejść w TTS opracowano mechanizm unikalnych identyfikatorów (UI), który przypisuje każdemu elementowi zadania unikalną liczbę (identyfikator) w taki sposób, że nie będzie ona współdzielona z innym elementem zadania. Korzystając z UI

oraz wybranej polityki zachowania można określić, które elementy zadania biorą udział w poszczególnych czynnościach reprezentowanych przez funkcje postępu misji. Ponadto unikalne identyfikatory pozwalają sprawdzić, czy zaplanowane do wykonania w tym samym przedziale czasu czynności są od siebie niezależne.

Algorytm weryfikacji modeli MAS opracowanych przy użyciu metodyki opisanej w rozdziale 3 i zdefiniowany w dalszej części tego rozdziału został opublikowany w [28], a także zaimplementowany [24].

4.2.1 Etap 4

Etap 4 algorytmu do weryfikacji modelu odpowiada za (częściową) weryfikację modelu oraz rozbudowę stanu scenariusza w zadanym momencie czasu.

Parametrami wejściowymi algorytmu etapu 4 są:

- aktualnie konstruowany stan scenariusza (w postaci dwugrafu);
- bijekcja przypisująca identyfikatory wierzchołków aktualnie skonstruowanego stanu do unikalnych identyfikatorów elementów zadania;
- czynność, która miała miejsce do momentu, w którym konstruowany jest stan systemu (w postaci reguły reakcji);
- bijekcja przypisująca identyfikatory wierzchołków redeksu do unikalnych identyfikatorów elementów zadania;
- bijekcja przypisująca wierzchołki stanu scenariusza w poprzednim momencie czasu do unikalnych identyfikatorów elementów zadania.

Wynikiem etapu 4 są:

- wariant 1 - model jest poprawny:
  - nowo skonstruowany stan systemu(dwugraf);
  - funkcja przypisująca wierzchołki nowo skonstruowanego stanu do unikalnych identyfikatorów;
  - wartość pierwszego nowego (dostępnego) unikalnego identyfikatora.
- wariant 2 - model jest niepoprawny:

– informacja o tym, której operacji nie udało się wykonać. Może to być operacja zastosowania reguły reakcji do stanu w poprzednim momencie czasu lub do aktualnie konstruowanego (podanego na wejściu).

Definicje formalne:

- $\mathcal{E} \subseteq \mathbb{N}$ - zbiór unikalnych identyfikatorów (UI) przypisywanych elementom zadania. Służy do „podążania" za obiektami pomiędzy stanami systemu.
- $Corr_{Red} : \mathcal{R} \to Red$ - funkcja zwracająca redeks podanej reguły reakcji.
- $M_{\mathcal{E}} \subset \{\mathcal{E} \to V_b \mid b \in Agt \cup Red\}$ - zbiór funkcji przypisujących do UI wierzchołki dwugrafów, którymi mogą być zarówno stany systemu (Agt), jak i wszystkie redeksy (Red).
- $IsUpdatePossible : Agt \times M_{\mathcal{E}} \times Red \times M_{\mathcal{E}} \to \{true, false\}$ - funkcja określająca, czy możliwe jest „zastosowanie" reguły reakcji do podanego na wejście stanu sytemu. Brane są pod uwagę mapowania UI na wierzchołki dwugrafu podanego na wejście oraz UI na wierzchołki podanego redeksu.
- $Update : Agt \times M_{\mathcal{E}} \times \mathcal{R} \times M_{\mathcal{E}} \times \mathcal{E} \to Agt \times M_{\mathcal{E}} \times \mathcal{E}$ - funkcja przekształcająca podany stan systemu. Polega na zastosowaniu na nim reguły reakcji (z uwzględnieniem mapowania UI na jego wierzchołki) oraz określeniu pierwszego nowego UI (w przypadku utworzenia nowych elementów otoczenia, którym zostanie przypisany unikalny identyfikator).

Mając wprowadzone powyższe definicje formalne można przejść do zdefiniowania sposobu działania etapu 4. Schemat blokowy algorytmu, w oparciu o który 4 etap sprawdza poprawność oraz rozszerza konstruowany stan został przedstawiony na rysunku 4.2. Parametry wejściowe tego algorytmu zostały opisane w tabeli 4.1, a wynik w tabeli 4.2. Celem tego etapu jest rozbudowa stanu scenariusza w postaci dwugrafu w taki sposób, aby uwzględniał wykonanie czynności reprezentowanej przez podaną na wejściu regułę reakcji. Aby wspomniana modyfikacja stanu systemu była poprawna, musi być wykonalna zarówno na dwugrafie reprezentującym stan scenariusza w poprzednim momencie czasu, jak i dotychczas skonstruowanej postaci stanu scenariusza dla docelowego momentu czasu. Tym sposobem sprawdzamy niezależność od siebie wykonywanych czynności.

Tabela 4.1: Parametry wejściowe 4 etapu algorytmu do weryfikacji modelu.

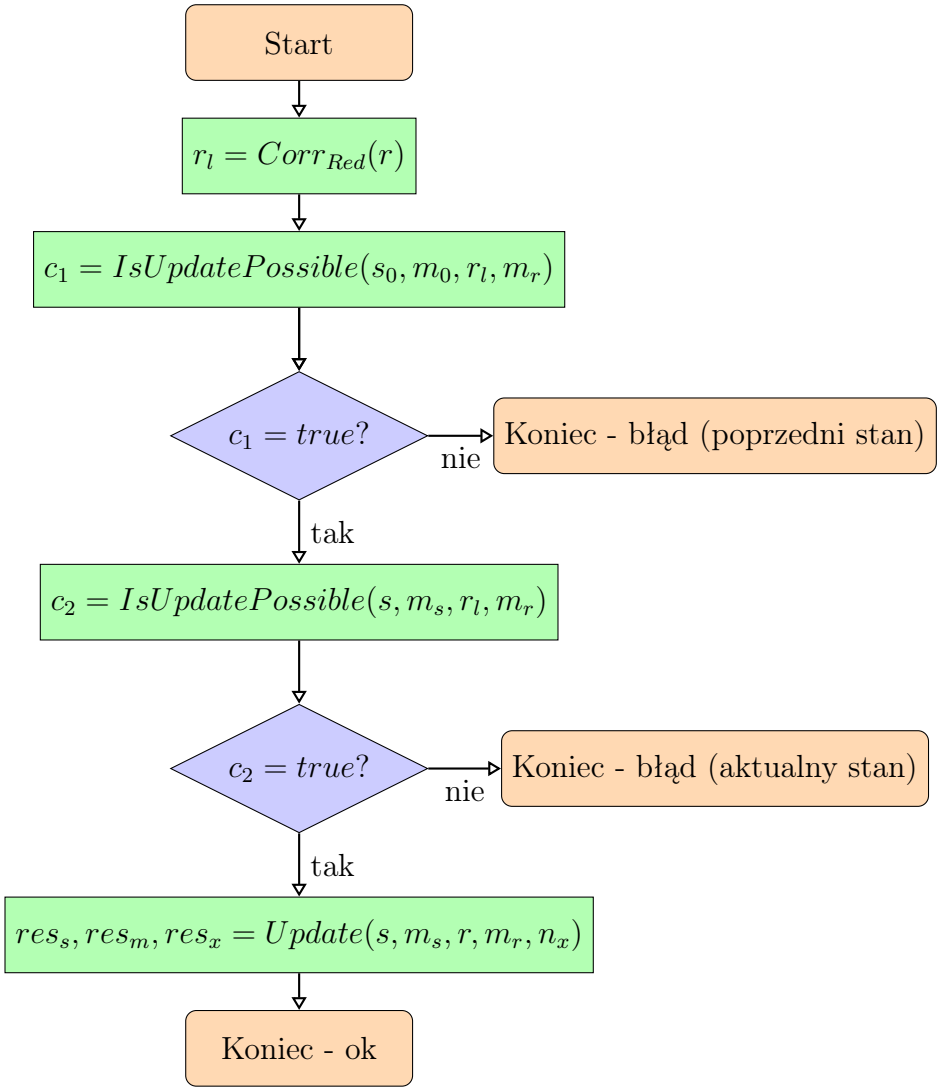
Zmienna	**Opis**
$s \in Agt$	Aktualnie skonstruowana postać stanu scenariusza w docelowym momencie czasu.
$m_s \in M_{\mathcal{E}}$	Funkcja przypisująca do UI wierzchołki s.
$r \in \mathcal{R}$	Reguła reakcji odpowiadająca czynności, która miała zostać wykonana do docelowego momentu czasu.
$m_r \in M_{\mathcal{E}}$	Funkcja przypisująca do UI wierzchołki redeksu reguły r.
$s_0 \in Agt$	Stan scenariusza w poprzednim momencie czasu.
$m_0 \in M_{\mathcal{E}}$	Mapowanie UI na wierzchołki s_0.
$n_x \in \mathcal{E}$	Pierwszy dostępny unikalny identyfikator, który może zostać przypisany elementom otoczenia powstałym w wyniku czynności reprezentowanej przez r.

Tabela 4.2: Wynik 4 etapu algorytmu do weryfikacji modelu.

Zmienna	**Opis**
$res_s \in Agt$	Aktualnie skonstruowany stan scenariusza w docelowym momencie czasu, który uwzględnia wykonanie czynności reprezentowanej przez r.
$res_m \in M_{\mathcal{E}}$	Funkcja przypisująca do UI wierzchołki dwugrafu res_s.
$res_x \in \mathcal{E}$	Pierwszy unikalny identyfikator dostępny po wykonaniu tej iteracji etapu 4.

Analizując schemat blokowy tego etapu można opisać go następująco. Pierwszym krokiem jest sprawdzenie możliwości zastosowania reguły reakcji do stanu w poprzednim momencie czasu, uwzględniając mapowanie UI na wystąpienie redeksu w tym dwugrafie. Jeżeli wynik tego testu jest pozytywny to wykonywany jest kolejny test. Tym razem sprawdzane jest zastosowanie tej samej reguły reakcji do aktualnej postaci stanu w docelowym momencie czasu. Jeżeli i ten test zakończy się pozytywnie, to wykonywana jest transformacja aktualnej postaci stanu w docelowym momencie czasu. Wynik tej transformacji jest rezultatem tego etapu algorytmu. Należy tutaj podkreślić, że ten etap jest wykonywany dla każdego elementu marszruty będącej polityką zachowania.

Jako podsumowanie przedstawione zostaną przykładowe dane wejściowe wraz z rezultatem opisanego etapu. W kolumnach tabeli 4.3 zamieszczono argumenty



Rysunek 4.2: Schemat blokowy 4 etapu algorytmu weryfikacji modelu MAS.

algorytmu etapu 4, w kolejności zgodnej z tą przedstawioną w tabeli 4.1. Wynik dla każdego zestawu danych ograniczono do stwierdzenia, czy wystąpił błąd i jeżeli tak, to co jest jego przyczyną.

W związku z tym, że funkcja *IsUpdatePossible* sprawdza, czy jest możliwe wykonanie funkcji *Update*, przygotowano szczegółowy przebieg tylko tej drugiej. Został on zamieszczony w załączniku D.

4.2.2 Etap 3

Etap 3 algorytmu do weryfikacji modelu odpowiada za konstrukcję stanu scenariusza w zadanym momencie czasu.

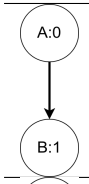
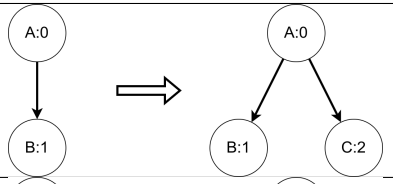
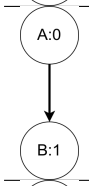
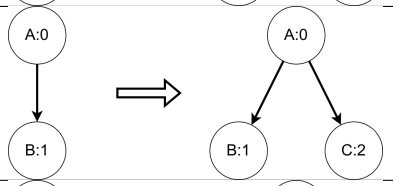
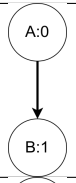
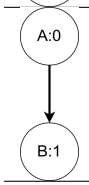
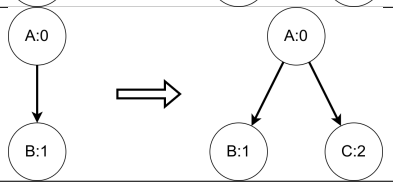
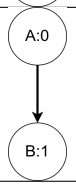
Parametrami wejściowymi etapu 3 są:

- stan scenariusza w poprzednim momencie czasu - w postaci dwugrafu;
- bijekcja przypisująca do unikalnych identyfikatorów wierzchołki powyższego stanu;
- podzbiór elementów rozszerzonej marszruty, które można wykorzystać do konstrukcji stanu scenariusza. Definicja marszruty rozszerzonej oraz kryteriów, przy użyciu których elementy tego zbioru są używane opisano poniżej;
- konfiguracja SaT scenariusza w poprzednim momencie czasu;
- moment czasu, dla którego stan scenariusza jest konstruowany;
- liczba obiektów w zadaniu.

Wynikiem etapu 3 są:

- wariant 1 - model jest poprawny:
  - podzbiór elementów rozszerzonej marszruty (podanej na wejściu), który nie został wykorzystany do konstrukcji stanu;
  - stan scenariusza w zadanym momencie czasu (dwugraf);
  - mapowanie unikalnych identyfikatorów na wierzchołki powyższego dwugrafu;
  - konfiguracja SaT skonstruowanego stanu;

Tabela 4.3: Przykładowe dane wejściowe dla algorytmu 4 etapu wraz z opisem rezultatu dla tych danych. Pierwszy zestaw danych wejściowych skutkuje błędem, ponieważ stan w poprzednim momencie czasu nie zawiera wystąpienia redeksu. Dla drugiego zestawu danych redeks znajduje się zarówno w dwugrafach s jak i *s0*. Jednak nie istnieje takie wystąpienie redeksu w dwugrafie s, dla którego mapowania m_r oraz m_s przypisują ten sam unikalny identyfikator.

s	m_s	r	m_r	s_0	m_0	n_x	Wynik
	$\{(0,0),(1,1)\}$		$\{(0,0),(1,1)\}$		$\{(0,0)\}$	3	Błąd (poprzedni stan)
	$\{(0,0),(1,2)\}$		$\{(0,0),(1,1)\}$		$\{(0,0),(1,1)\}$	3	Błąd (aktualny stan)
	$\{(0,0),(1,1)\}$		$\{(0,0),(1,1)\}$		$\{(0,0),(1,1)\}$	3	Ok

  - fragment harmonogramu działań dla elementów zdania, dla których określono czynności do wykonania, kończące się w zadanym momencie czasu.

- wariant 2 - model jest niepoprawny:
  - stan systemu będący rozwinięciem stanu scenariusza w poprzednim momencie czasu, wraz z mapowaniem UI na jego wierzchołki, który miał być poddany (nieudanej) transformacji w ramach etapu 4;
  - stan scenariusza w poprzednim momencie czasu, wraz z mapowaniem UI na jego wierzchołki, który miał być poddany nieudanej transformacji w ramach etapu 4;
  - reguła reakcji wraz mapowaniem UI na jej redeks.

Definicje formalne:

- $A \subseteq 2^{\mathbb{N}_+}$ - zbiór potęgowy identyfikatorów obiektów zadania. Te same identyfikatory są używane w konfiguracjach SaT.
- $W_M \subset \mathbb{N} \times T \times M_{\mathcal{E}} \times M_{\mathcal{E}} \times \mathcal{E} \times (A \times \mathbb{N})$ - rozszerzona marszruta składająca się z:
  1. liczby pozycyjnej, który element marszruty jest rozpatrywany;
  2. funkcji postępu misji;
  3. funkcji przypisującej do UI wierzchołki redeksu. Redeks należy do reguły reakcji użytej przy konstrukcji przejścia w TTS, z którego powstała powyższa funkcja postępu;
  4. funkcji przypisującej do UI wierzchołki stanu będącego rezultatem przejścia w TTS, z którego powstała powyższa funkcja postępu misji;
  5. pierwszego nowego UI, który można przypisać do nowego elementu otoczenia powstałego w wyniku zastosowania reguły reakcji;
  6. zbioru identyfikatorów obiektów zadania (używanych w SaT), które biorą udział czynności utożsamianej z funkcją postępu misji wraz z czasem trwania tej czynności.

- $First_M : 2^{W_M} \rightarrow W_M \times 2^{W_M}$ - funkcja zwracająca najmniejszy element w podzbiorze elementów marszruty rozszerzonej oraz marszrutę skróconą o ten element.

- $Corr_{Tra} : T \rightarrow Tra$ - funkcja przypisująca przejścia w TTS do funkcji postępu misji.

- $Objects_U : I \times (A \times \mathbb{N}) \rightarrow I$ - funkcja aktualizująca konfigurację SaT. Przyjmuje jako argument aktualną konfigurację, zbiór identyfikatorów obiektów, które będą aktualizowane oraz czas o jaki nastąpi aktualizacja. Wynikiem jest nowa konfiguracja SaT.
  Przykład działania ww. funkcji:

$$Objects_U(\langle(1,15),(2,10),(3,7)\rangle, \langle(\{1,3\},2)\rangle) = \langle(1,17),(2,10),(3,9)\rangle$$

- $Objects_F : I \times \mathbb{N} \rightarrow A$ - funkcja określająca, dla których obiektów zadania zaplanowane są czynności później niż aktualnie konstruowany stan. Przyjmuje konfigurację SaT i moment czasu, dla którego jest konstruowany stan. Przykład działania:

$$Objects_F(\langle(1,15),(2,10),(3,7)\rangle, 10) = \{1\}$$

- $Corr_{\mathcal{R}} : Tra \rightarrow \mathcal{R}$ - funkcja przypisująca reguły reakcji do przejść w TTS.

- $Plan : T \times M_{\mathcal{E}} \times \mathcal{R} \times \mathbb{N} \times \mathbb{N} \rightarrow Schedule$ - funkcja konstruująca fragment harmonogramu działań dla elementów zadania. Przyjmuje jako argument funkcję postępu misji, mapowanie UI na wierzchołki redeksu, regułę reakcji, moment czasu, dla którego konstruowany jest stan scenariusza oraz ilość czasu potrzebną na wykonanie czynności odpowiadającej funkcji przejścia. Ma ona następującą definicję:

$$Plan(f, m_r, r, d, n_d) = \begin{pmatrix} Assignment = dom(m_r) \times \{(f, lab(r), d - n_d)\}, \\ role = \{(\ (e, (f, lab(r), d - n_d)), m_r(e)\ ) \mid \forall e \in dom(m_r)\}, \\ duration = (f, n_d) \end{pmatrix}$$

Tabela 4.4: Parametry wejściowe 3 etapu algorytmu do weryfikacji modelu.

Zmienna	**Opis**
$s_0 \in Agt$	Stan scenariusza w poprzednim momencie czasu.
$m_0 \in M_{\mathcal{E}}$	Bijekcja przypisująca do UI wierzchołki s_0.
$W \subset W_M$	Elementy rozszerzonej marszruty, które mogą zostać wykorzystane do konstrukcji stanu scenariusza.
$i_0 \in I$	Konfiguracja SaT stanu w poprzednim momencie czasu.
$d \in \mathbb{N}$	Moment czasu, dla którego konstruowany będzie stan scenariusza.
$n_o \in \mathbb{N}$	Liczba obiektów w zadaniu.

Funkcja $lab : \mathcal{R} \to Lab$ zwraca etykietę reguły reakcji podanej na wejściu.

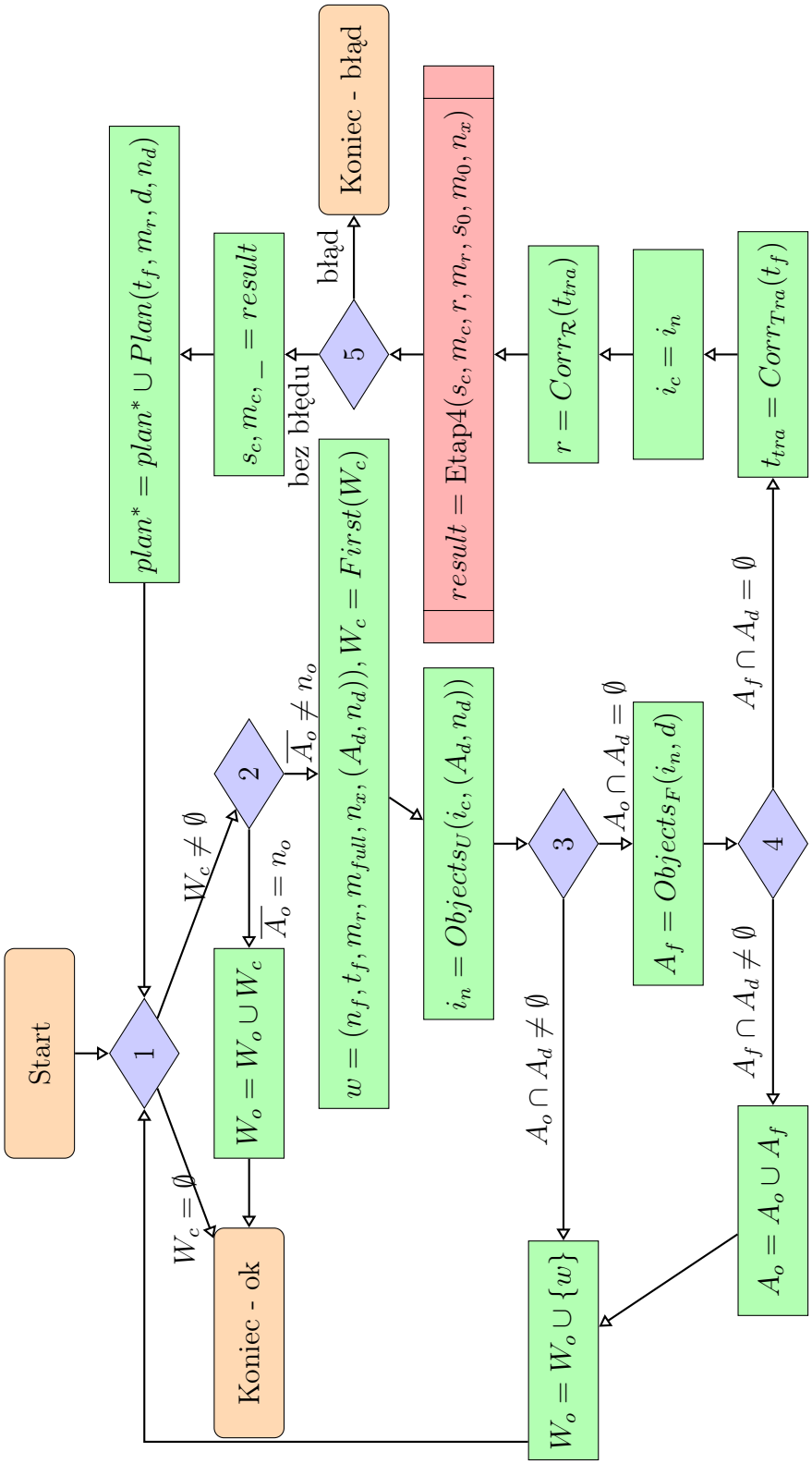
Wykorzystując powyższe definicje można przejść do algorytmu etapu 3. Rysunek 4.3 przedstawia schemat blokowy tego etapu. W blokach decyzyjnych na schemacie umieszczono numery odpowiadające poniżej zdefiniowanym instrukcjom warunkowym.

1. Pierwsza instrukcja warunkowa sprawdza, czy osiągnięty został koniec marszruty. Jeżeli tak, to mamy pewność, że aktualnie skonstruowany stan odpowiada stanowi w zadanym momencie czasu.

2. W tym teście jest sprawdzane, czy dla któregokolwiek obiektu zadania można jeszcze zaplanować jakąś czynność tak, aby kończyła się w momencie czasu d. Jeżeli nie, to mamy pewność, że aktualna postać stanu scenariusza odpowiada stanowi w zadanym momencie czasu.

3. Czy którykolwiek z obiektów zaangażowanych w aktualną czynność należy do zbioru obiektów pomijanych? Jeżeli tak, to ten element marszruty jest pomijany.

4. Czy obiekty biorące udział w aktualnie rozpatrywanej czynności ukończą ją do chwili d? Jeżeli nie, to pomijamy tę czynność w konstrukcji stanu i dodajemy wszystkie zaangażowane w nią obiekty do zbioru pomijanych obiektów zadania.



Rysunek 4.3: Schemat blokowy 3 etapu algorytmu weryfikacji modelu MAS.

Tabela 4.5: Zmienne pomocnicze używane w trakcie wykonywania etapu 3.

Zmienna	**Opis**
$s_c \in Agt$	Aktualnie skonstruowana postać stanu scenariusza w momencie d. Wartość początkowa to s_0.
$m_c \in M_{\mathcal{E}}$	Funkcja przypisująca do UI wierzchołki s_c. Wartość początkowa to m_0
$i_c \in I$	Konfiguracja SaT aktualnie konstruowanego stanu. Wartość początkowa to i_0.
$A_o \in A$	Zbiór pomijanych obiektów zadania. Dla wszystkich obiektów z tego zbioru zaplanowano czynności kończące się nie później niż w momencie d. Wartość początkowa to zbiór pusty.
$W_c \subseteq W$	Zbiór elementów marszruty podanej na wejściu, które mogą być nadal wykorzystane do konstrukcji stanu scenariusza w momencie d. Wartość początkowa to W.
$W_o \subseteq W$	Zbiór elementów marszruty, które nie mogą być już wykorzystane. Wartość początkowa to zbiór pusty.
$plan^* \in Schedule$	Fragment harmonogramu działań zawierający czynności, które skończą się w momencie czasu d. Początkowa wartość to $(\emptyset, \emptyset, \emptyset)$.

Tabela 4.6: Wynik 3 etapu algorytmu do weryfikacji modelu.

Zmienna	**Opis**
$W_o \subset W$	Elementy marszruty podanej na wejściu, które nie zostały wykorzystane do konstrukcji stanu.
$s_c \in Agt$	Stan scenariusza w momencie d.
$m_c \in M_x$	Funkcja przypisująca do UI wierzchołki s_c.
$i_c \in I$	Konfiguracja SaT stanu w momencie d.
$plan^* \in Schedule$	Fragment harmonogramu dla elementów zadania, dla których określono czynności do wykonania kończące się w momencie czasu d.

5. Jeżeli etap 4 zakończył się z błędem, to model jest niepoprawny i wykonywanie etapu 3 zostaje przerwane.

W tabelach 4.4 i 4.6 omówiono odpowiednio parametry wejściowe i rezultaty algorytmu, zaś w tabeli 4.5 przedstawiono zmienne pomocnicze używane w trakcie jego wykonywania.

Sposób działania 3 etapu można opisać następująco. Każdej funkcji postępu misji będącej elementem marszruty odpowiada któreś z przejść w TTS. Tym z kolei odpowiadają reguły reakcji. Zaczynając od stanu scenariusza w poprzednim momencie czasu dokonywana jest próba zastosowania kolejnych reguł reakcji, odpowiadających kolejnym funkcjom postępu misji. Elementy biorące udział w reakcji (czynności) są wskazywane przez funkcję m_r, będącą składnikiem każdego elementu marszruty rozszerzonej. Jeżeli czynność odpowiadająca rozpatrywanej regule reakcji zakończy się do chwili, dla której konstruowany jest stan scenariusza, oraz żaden zaangażowany w nią obiekt zadania nie jest pomijany, to następuje przejście do etapu 4. W przypadku, gdy rozpatrywana czynność miałaby zakończyć się później niż w momencie d, to element marszruty, który ją zawiera jest pomijany. Wszystkie pominięte elementy marszruty mogą być wykorzystane w kolejnych iteracjach etapu 3, których celem jest konstrukcja stanu scenariusza w późniejszym momencie czasu. Ponadto wszystkie obiekty zadania, które miały brać udział w pomijanej czynności są oznaczane jako „zajęte", a każda następna czynność na marszrucie angażująca dowolny z tych obiektów będzie pomijana. Dwugraf s_c w trakcie etapu 3 jest stanem systemu, a nie stanem scenariusza. Dopiero po zakończeniu tego etapu odzwierciedla on sytuację, w jakiej będzie system w trakcie wykonywania zadania, przy użyciu opracowanej polityki zachowania.

Ze względu na złożoność oraz objętość etapu 3 pominięto tutaj jego przykładowy przebieg. Szczegółowe przebiegi etapu 3 zamieszczono w rozdziałach 5.2 i 5.3, w których przedstawiono zarówno proces konstrukcji stanów scenariuszy w poprawnych modelach, jak i niepoprawnych.

4.2.3 Etap 2

Etap 2 algorytmu weryfikacji modelu odpowiada za rozszerzenie marszruty do postaci akceptowalnej przez etap 3.

Parametrami wejściowymi algorytmu etapu 2 są:

- marszruta w przestrzeni stanów;
- liczba obiektów w zadaniu.

Wynikiem etapu 2 jest:

- rozszerzona marszruta.

Definicje formalne:

- $W \subseteq \mathbb{N} \times T$ - marszruta w przestrzeni stanów.
- $First : 2^W \to W \times 2^W$ - funkcja zwracająca „najmniejszy" element marszruty oraz marszrutę skróconą o ten właśnie element.
- $Trans : Tra \times M_{\mathcal{E}} \times \mathcal{E} \to M_{\mathcal{E}} \times M_{\mathcal{E}} \times \mathcal{E}$ - funkcja przekształcająca mapowanie unikalnych identyfikatorów. Jako argumenty wejściowe przyjmuje kolejno:
  1. przejście w TTS, z którego powstała funkcja postępu misji, będąca elementem przekształcanej marszruty;
  2. aktualną funkcję przypisującą wierzchołki do unikalnych identyfikatorów;
  3. pierwszy dostępny (najmniejszy) unikalny identyfikator, któremu może zostać przypisany wierzchołek, w przypadku powstania nowych elementów otoczenia.

  Wynikiem tej funkcji jest trójka uporządkowana, której elementami są kolejno:

  1. funkcja przypisująca wierzchołki redeksu reguły reakcji powiązanej z podanym na wejściu przejściem, do unikalnych identyfikatorów;
  2. funkcja przypisująca wierzchołki stanu wynikowego przejścia podanego jako argument, do unikalnych identyfikatorów;
  3. nowy najmniejszy UI, któremu może zostać przypisany jakiś wierzchołek.

Tabela 4.7: Parametry wejściowe algorytmu etapu 2.

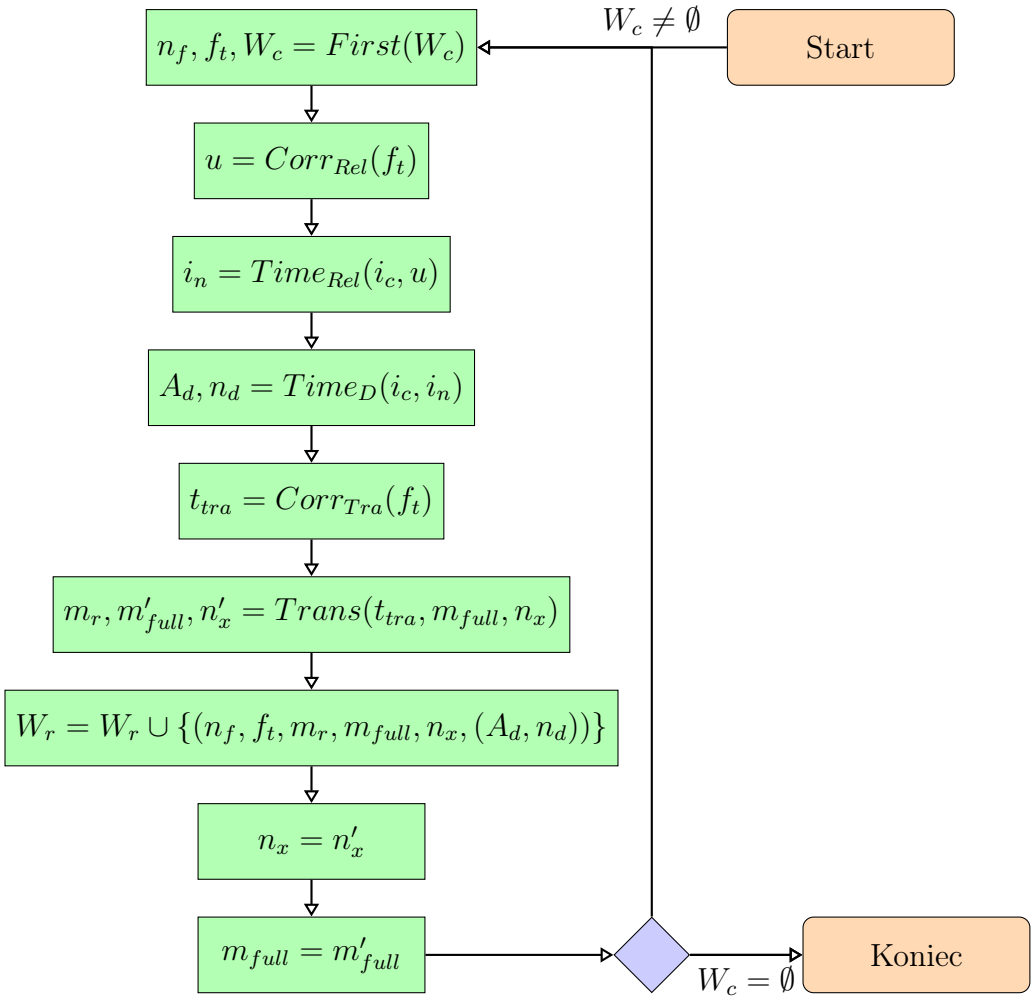
Zmienna	**Opis**
W	Marszruta w przestrzeni stanów.
n_o	Liczba obiektów w zadaniu.

- $Rel \subseteq \mathbb{N}_1^2 \times \cdots \times \mathbb{N}_{n_o}^2$ - zbiór wskazujący na względną zmianę w konfiguracji SaT.
- $Corr_{Rel} : T \to Rel$ - funkcja przekształcająca funkcje postępu misji we względne zmiany SaT.
- $Time_{Rel} : I \times Rel \to I$ - funkcja aktualizująca konfigurację SaT.
- $Time_D : I \times I \to A \times \mathbb{N}$ - funkcja różnicy czasowej dla obiektów zadania pomiędzy dwiema SaT konfiguracjami. Zwraca zbiór obiektów zaangażowanych w czynność oraz czas jej trwania.

Mając powyżej wprowadzone definicje formalne można przystąpić do zdefiniowania algorytmu etapu 2. Polega on na rozszerzaniu kolejnych elementów marszruty w przestrzeni stanów. Operacja rozszerzania ma na celu określenie, które unikalne obiekty (nie należy mylić z obiektami zadania) biorą udział w kolejnych czynnościach, reprezentowanych przez funkcje postępu misji. Rozszerzanie jest wykonywane w kolejności elementów marszruty podanej jako parametr wejściowy. Etap ten jest wykonywany tylko raz, przed przystąpieniem po raz pierwszy do wykonywania etapu 3.

Schemat blokowy algorytmu etapu 2 zamieszczono na rysunku 4.4. Parametry wejściowe tego etapu przedstawiono w tabeli 4.7, a zmienne pomocnicze wraz z wynikiem omówiono w tabeli 4.8. Rozszerzenie marszruty wejściowej zachowuje kolejność jej elementów.

Jako przykład oczekiwanego wyniku etapu 2 wykorzystane będą marszruty utworzone z TTS z tabeli 2.2. Wszystkie funkcje postępu misji skonstruowane na podstawie przejść z tego TTS zostały przedstawione w tabeli 3.1. Danymi wejściowymi będą marszruty $\{(1, f_1), (2, f_4)\}$ i $\{(1, f_2), (2, f_4)\}$. Porównanie obydwu rozszerzonych marszrut pozwoli podkreślić szczegóły wyniku procesu rozszerzania, bez zagłębiania się w jego przebieg. W tabeli 4.9 przedstawiono roz-



Rysunek 4.4: Schemat blokowy etapu 2 algorytmu weryfikacji modelu MAS.

Tabela 4.8: Zmienne pomocnicze używane w trakcie wykonywania etapu 2 wraz z jego wynikiem.

Zmienna	**Opis**
$n_x \in \mathcal{E}$	Wartość pierwszego nowego unikalnego identyfikatora. Wartością początkową jest liczba wierzchołków stanu wejściowego przejścia w TTS, z którego skonstruowano pierwszy element marszruty.
$m_{full} \in M_{\mathcal{E}}$	Aktualne mapowanie wierzchołków ostatnio rozpatrywanego stanu wyjściowego na UI. Ostatnio rozpatrywanym stanem wyjściowym jest stan wyjściowy przejścia w TTS, które odpowiada ostatnio rozszerzanej funkcji postępu misji. Wartością początkową tej zmiennej jest funkcja przypisująca kolejne wierzchołki (ich identyfikatory) stanu wejściowego przejścia z TTS, odpowiadającemu pierwszemu elementowi marszruty do kolejnych liczb naturalnych (unikalnych identyfikatorów).
$W_r \subset W_M$	Elementy rozszerzonej marszruty. Wartością początkową jest zbiór pusty. **Zbiór ten jest wynikiem etapu 2**.
$W_c \subset W$	Zbiór elementów marszruty podanej na wejściu, które nie zostały jeszcze rozszerzone. Wartością początkową jest zbiór W.
$i_c \in I$	Aktualna konfiguracja SaT. Wartością początkową jest krotka $((1,0),\ldots,(n_o,0))$.

szerzone elementy obydwu marszrut. Możemy z ich wykorzystaniem odczytać, które unikalne obiekty biorą udział w kolejnych czynnościach (reprezentowanych przez reguły reakcji) oraz jakie jest rozmieszczenie unikalnych obiektów, w stanach systemu poddawanych wskazanym przez funkcje postępu misji czynnościom. Należy tutaj podkreślić, że kolejne dwugrafy będące stanami wynikowymi przejść w TTS, które odpowiadają funkcjom postępu misji w marszrucie, nie stanowią reprezentacji kolejnych stanów scenariusza.

Różnicą w przebiegu misji, którą można odczytać porównując obydwie rozszerzone marszruty jest kolejność w jakiej (unikalne) obiekty przemieszczają się z obszaru A do obszaru B. W marszrucie $\{(1, f_1), (2, f_4)\}$ pierwszym obiektem jest ten o UI równym 1, co można odczytać z mapowania UI na wierzchołek o identyfikatorze 1 w redeksie. Dla drugiej marszruty pierwszym przechodzącym unikalnym obiektem jest ten o identyfikatorze 2. Funkcja f_4 w obydwu marszrutach „przesuwa" drugi (pozostały) obiekt typu U z obszaru typu A do B, którym w przypadku pierwszej marszruty jest obiekt o UI równym 2, a dla drugiej marszruty równym 1.

Zasadnym jest pytanie, co zawiera rozszerzona marszruta, czego nie można odczytać bezpośrednio z elementów źródłowego Śledzącego Systemu Przejść. Każdy element TTS umożliwia analizę czasową tylko pojedynczej czynności. Oznacza to, że mając do dyspozycji tylko jedno przejście w TTS wiemy, które wierzchołki stanu wejściowego odpowiadają wierzchołkom stanu wynikowego. Innymi słowy, analizując dowolną parę przejść w TTS nie można określić, czy te same (nie takie same) obiekty biorą udział w czynnościach. Z kolei, mając dowolny element rozszerzonej marszruty wiemy dokładnie, które obiekty biorą udział w czynności, reprezentowanej przez funkcję postępu misji. Unikalne identyfikatory mają na celu jednoznaczne rozróżnienie obiektów, bez względu na ich typ, moment pojawienia się lub rolę, jaką odgrywają w trakcie scenariusza.

Szczegółowy przebieg procesu rozszerzania marszruty dla zadania zdefiniowanego w rozdziale 5.2 został przedstawiony w załączniku F.

Tabela 4.9: Przykładowy wynik etapu 2. Danymi wejściowymi są dwie marszruty utworzone z TTS przedstawionego w tabeli 2.2. Definicje funkcji postępu misji znajdują się w tabeli 3.1.

$\mathbb{N}$	T	**UI na redeks**	**UI na stan wynikowy**	$\mathcal{E}$	$(A \times \mathbb{N})$
marszruta $\{(1, f_1), (2, f_4)\}$					
1	f_1	$\{(0,0),(1,1),(3,2)\}$	$\{(0,0),(1,1),(2,2),(3,3)\}$	4	$(\{1\}, 2)$
2	f_4	$\{(0,0),(2,1),(3,2)\}$	$\{(0,0),(1,3),(2,1),(3,2)\}$	4	$(\{2\}, 2)$
marszruta $\{(1, f_2), (2, f_4)\}$					
1	f_2	$\{(0,0),(2,1),(3,2)\}$	$\{(0,0),(1,1),(2,2),(3,3)\}$	4	$(\{2\}, 2)$
2	f_4	$\{(0,0),(1,1),(3,2)\}$	$\{(0,0),(1,1),(2,3),(3,2)\}$	4	$(\{1\}, 2)$

4.2.4 Etap 1

Ostatni omawiany etap algorytmu do weryfikacji modelu MAS, a jednocześnie pierwszy w kolejności wykonywania, odpowiada za weryfikację modelu oraz konstrukcję stanów scenariusza w kolejnych momentach czasu.

Parametrami wejściowymi algorytmu etapu 1 są:

- liczba obiektów w zadaniu;
- marszruta w przestrzeni stanów.

Wynik etapu 1 :

1. wariant 1 - model jest poprawny:
    - zbiór stanów scenariusza w kolejnych momentach czasu wraz z mapowaniami unikalnych identyfikatorów na wierzchołki tych stanów i ich konfiguracje SaT;
    - harmonogram działań (zdefiniowany w 3.4) dla wszystkich elementów zadania występujących w scenariuszu.
2. wariant 2 - model jest niepoprawny:
    - moment czasu, dla którego stan scenariusza w postaci dwugrafowej nie mógł zostać skonstruowany;
    - element, który nie mógł zostać poddany transformacji w trakcie etapu 4;
    - reguła reakcji, która miała zostać użyta do (nieudanej) transformacji;

Tabela 4.10: Parametry wejściowe algorytmu etapu 1.

Zmienna	**Opis**
W	Marszruta w przestrzeni stanów.
n_o	Liczba obiektów zadania.

- mapowania UI na wierzchołki stanu, który nie mógł zostać poddany transformacji na etapie 4 oraz na wierzchołki powyższej reguły reakcji.

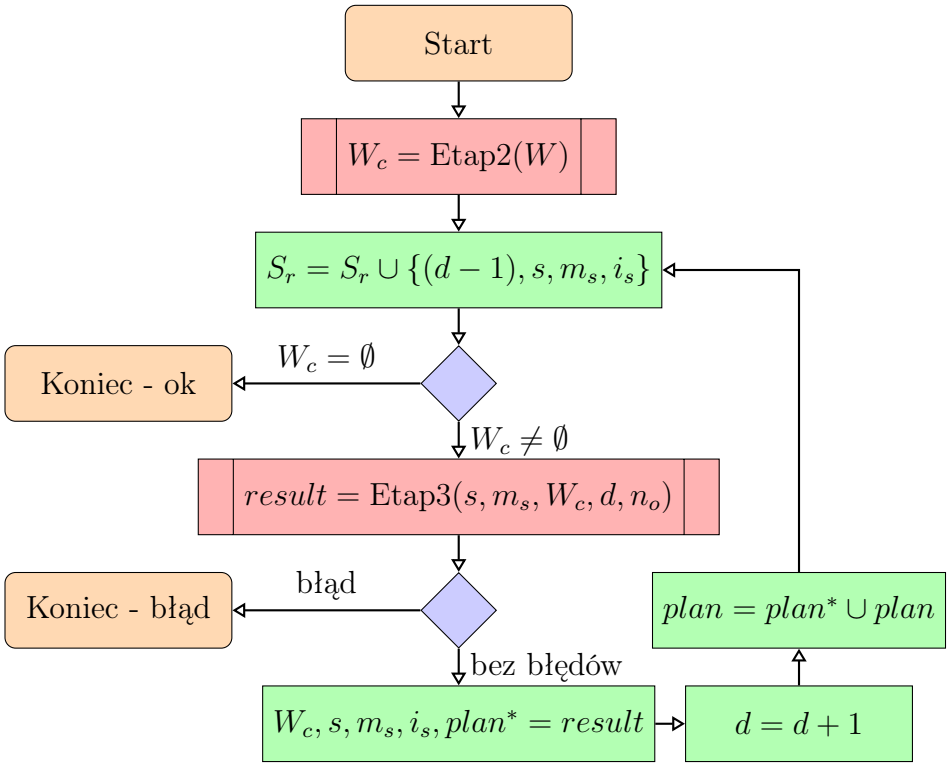
Weryfikacja modeli MAS, wykonywana z użyciem omawianego w tym rozdziale algorytmu polega na sprawdzeniu, czy wybrana polityka zachowania umożliwia opracowanie harmonogramu działań dla elementów zadania (w szczególności agentów w systemie). Innymi słowy sprawdzane jest, czy można opracować scenariusz na podstawie wybranej polityki zachowania. Należy tu jeszcze raz podkreślić różnicę pomiędzy stanem systemu, czyli dwugrafem ze zbioru *Agt*, a stanem scenariusza, czyli dwugrafem reprezentującym stan systemu w jakimś momencie czasu, w trakcie wykonywania scenariusza.
Proces weryfikacji rozpoczyna się od etapu 1. Wykonuje on rozszerzenie polityki zachowania (czyli marszruty w przestrzeni stanów), w sposób opisany w poprzednim rozdziale, a następnie konstruowane są stany scenariusza w postaci dwugrafów, w kolejnych momentach czasu. Każdy taki dwugraf stanowi część wyniku opracowanego algorytmu. Kolejne stany są konstruowane dopóki nie zostanie „wykorzystana" cała (rozszerzona) marszruta. Przebieg etapu 1 przedstawiono na rysunku 4.5. Parametry wejściowe algorytmu etapu 1 zdefiniowano w tabeli 4.10, a zmienne pomocnicze wraz z wynikiem zostały przedstawione odpowiednio w tabelach 4.11 i 4.12.

Tabela 4.11: Zmienne pomocnicze algorytmu etapu 1.

Zmienna	**Opis**
$W_c \subset W_M$	Zbiór elementów rozszerzonej marszruty, które nie zostały jeszcze wykorzystane. Wartością początkową jest zbiór pusty, ale faktyczna inicjalizacja tej zmiennej następuje dopiero z wykorzystaniem wyniku etapu 2.
d	Aktualny moment czasu, dla którego konstruowany jest stan scenariusza. Początkowa wartość to 1.
$s \in Agt$	Stan scenariusza w momencie czasu $d-1$. Początkowa wartość to stan wejściowy przejścia w TTS, odpowiadający pierwszemu elementowi marszruty W.
$m_s \in M_{\mathcal{E}}$	Mapowanie UI na wierzchołki stanu s. Początkowa wartość to bijekcja kolejnych liczb naturalnych na wierzchołki s.
$i_s \in I$	Konfiguracja SaT stanu scenariusza w momencie $d-1$. Początkowa wartość to $((1,0),\ldots,(n_o,0))$.
$S_r \subset \mathbb{N} \times Agt \times M_{\mathcal{E}} \times I$	Zbiór stanów scenariusza wraz z odpowiadającymi im mapowaniami UI i SaT konfiguracjami w kolejnych momentach czasu. Początkowa wartość to zbiór pusty.
$plan^* \in Schedule$	Fragment harmonogramu działań zawierający czynności, które skończą się w momencie czasu d. Początkowa wartość to $(\emptyset,\emptyset,\emptyset)$.

Tabela 4.12: Wyniki etapu 1 w przypadku, gdy weryfikowany model jest poprawny.

Zmienna	**Opis**
$S_r \subset \mathbb{N} \times Agt \times M_{\mathcal{E}} \times I$	Reprezentacja formalna przebiegu scenariusza. Elementami składowymi są czwórki uporządkowane, których kolejne składowe oznaczają: moment czasu; stan scenariusza w postaci dwugrafu; funkcję przypisującą wierzchołki wspomnianego dwugrafu, unikalnym identyfikatorom reprezentującym elementy zadania; konfigurację SaT stanu scenariusza.
$plan \in Schedule$	Harmonogram działań dla wszystkich elementów zadania występujących w trakcie scenariusza. Początkowa wartość to $(\emptyset,\emptyset,\emptyset)$.



Rysunek 4.5: Schemat blokowy etapu 1 algorytmu weryfikacji modelu MAS.

Rozdział 5

Zastosowanie metodyki projektowania zachowania roju dronów

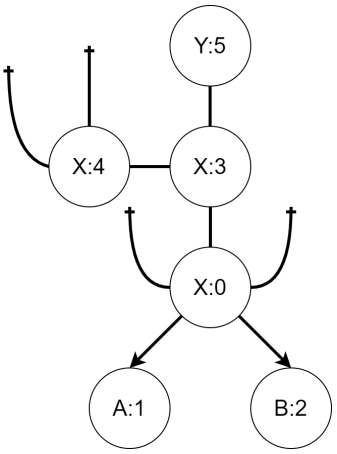
W tym rozdziale zostaną przedstawione szczegółowe przykłady zastosowania omówionej w poprzednich rozdziałach metodyki projektowania roju dronów oraz metody weryfikacji opracowanych modeli.

Każdy przykład będzie miał na celu przedstawienie użyteczności opracowanych elementów w wybranym obszarze. Większość przedstawionych przypadków użycia będzie poparta dokładnym przebiegiem opracowanych algorytmów, zarówno w ramach metodyki projektowania z rozdziału 3, jak i metody weryfikacji z rozdziału 4.2.

5.1 Działania kooperacyjne w trakcie misji

Pierwszy przykład ma na celu zademonstrowanie możliwości wykorzystania metodyki do opracowania sposobu zachowania dronów, realizujących zadanie możliwe do wykonania na kilka sposobów, które wymagają między nimi współpracy.

Zadanie na potrzeby przykładu 5.1 jest następujące:
Na mapie złożonej z dwóch typów obszarów znajdują się dwa drony różnych typów. Typy obszarów oznaczymy jako X i Y, a typy dronów przez A i B. Drony mogą przemieszczać się zarówno pomiędzy obszarami X oraz z X do Y. Prze-



Rysunek 5.1: Stan początkowy zadania, w którym wymagana jest współpraca między dronami.

mieszczanie pomiędzy obszarami typu X zajmuje dronom typu A 1 jednostkę czasu, a dronom typu B 3 jednostki. Przemieszczenie z obszaru X do obszaru Y wymaga współpracy pomiędzy dronami i zajmuje 4 jednostki czasu. Celem zadania jest przemieścić obydwa drony na obszar typu Y. Stan początkowy przedstawiono na rysunku 5.1.

Na podstawie powyższego opisu zadania można zidentyfikować trzy reguły reakcji, które zostały przedstawione w tabeli 5.1. Wykorzystując stan początkowy oraz wspomniane reguły można skonstruować sześć dwugrafów, które będą stanowić elementy zbioru $\mathcal{B}$ dla śledzącego dwugrafowego systemu reakcyjnego. Zbiór ten wraz z relacją reakcji został przedstawiony na rysunku 5.2.

Na podstawie tak zdefiniowanego TBRS można skonstruować system przejść jak w tabeli 5.2. Ten z kolei może zostać przekształcony do przestrzeni stanów przedstawionej na rysunku 5.3. Definicje funkcji postępu misji będące elementami składowymi tej przestrzeni stanów znajdują się w tabeli 5.3. W celu wyznaczenia polityki zachowania należy na ich podstawie skonstruować macierze $\mathbb{M}_F^t$ oraz $\mathbb{M}_K^0$.

Tabela 5.1: Elementy zbioru $\mathcal{R}$ TBRS dla przykładu 5.1 wymagającego współpracy pomiędzy dronami. Funkcje śledzące (τ) są tożsamościami.

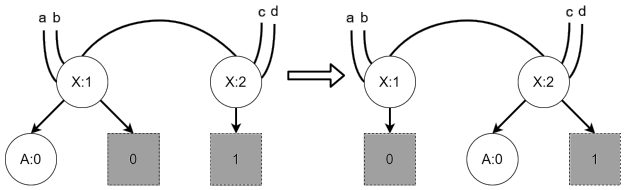
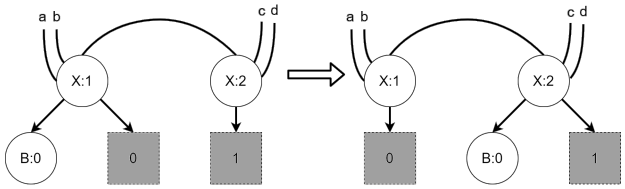
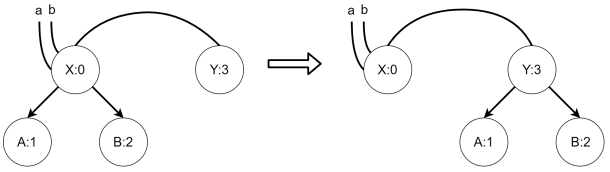
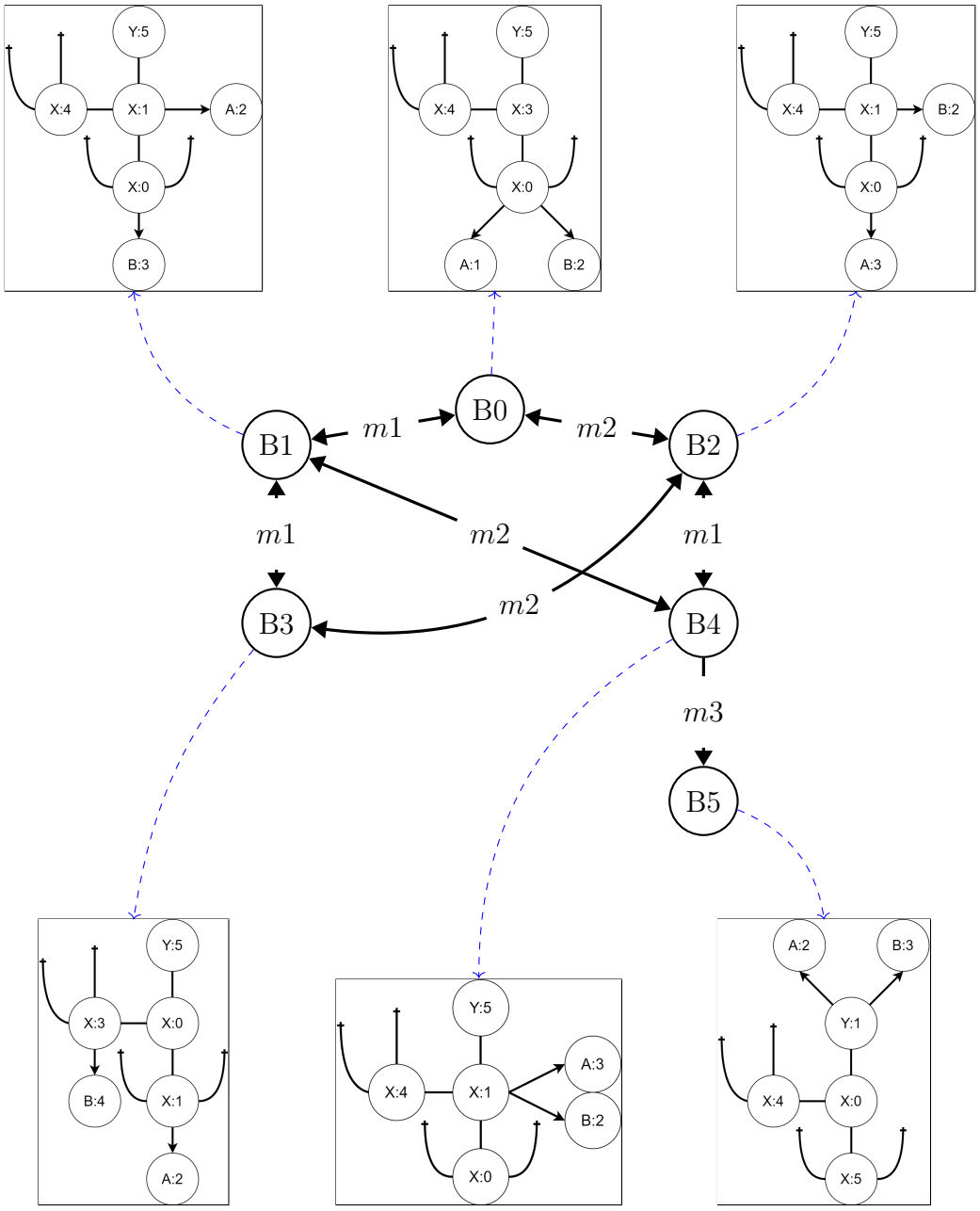
Nazwa reguły reakcji	Reprezentacja graficzna
$m1$	
$m2$	
$m3$	

Tabela 5.2: Śledzący system przejść dla przykładu zadania wymagającego współpracy między dronami. Każdy wiersz definiuje pojedyncze przejście.

Apl	Agt	Par	Res
$\langle B0, m1\rangle$	B1	$\{(0,0),(1,1),(2,3)\}$	$\{(0,0),(1,3),(2,1),(3,2),(4,4),(5,5)\}$
$\langle B0, m2\rangle$	B2	$\{(0,0),(1,2),(2,3)\}$	$\{(0,0),(1,3),(2,2),(3,1),(4,4),(5,5)\}$
$\langle B1, m1\rangle$	B3	$\{(0,1),(1,2),(2,4)\}$	$\{(0,1),(1,4),(2,2),(3,0),(4,3),(5,5)\}$
$\langle B1, m1\rangle$	B4	$\{(0,0),(1,3),(2,1)\}$	$\{(0,4),(1,1),(2,3),(3,2),(4,0),(5,5)\}$
$\langle B1, m1\rangle$	B0	$\{(0,1),(1,2),(2,0)\}$	$\{(0,0),(1,2),(2,3),(3,1),(4,4),(5,5)\}$
$\langle B2, m2\rangle$	B3	$\{(0,1),(1,2),(2,4)\}$	$\{(0,1),(1,0),(2,3),(3,4),(4,2),(5,5)\}$
$\langle B2, m1\rangle$	B4	$\{(0,0),(1,3),(2,1)\}$	$\{(0,4),(1,1),(2,2),(3,3),(4,0),(5,5)\}$
$\langle B2, m2\rangle$	B0	$\{(0,1),(1,2),(2,0)\}$	$\{(0,0),(1,3),(2,2),(3,1),(4,4),(5,5)\}$
$\langle B3, m2\rangle$	B2	$\{(0,3),(1,4),(2,0)\}$	$\{(0,1),(1,0),(2,4),(3,2),(4,3),(5,5)\}$
$\langle B3, m1\rangle$	B1	$\{(0,1),(1,2),(2,0)\}$	$\{(0,3),(1,0),(2,2),(3,4),(4,1),(5,5)\}$
$\langle B4, m3\rangle$	B5	$\{(0,1),(1,3),(2,2),(3,5)\}$	$\{(0,1),(1,5),(2,3),(3,2),(4,4),(5,0)\}$
$\langle B4, m1\rangle$	B2	$\{(0,1),(1,3),(2,4)\}$	$\{(0,4),(1,1),(2,2),(3,3),(4,0),(5,5)\}$
$\langle B4, m1\rangle$	B2	$\{(0,1),(1,3),(2,0)\}$	$\{(0,0),(1,1),(2,2),(3,3),(4,4),(5,5)\}$
$\langle B4, m2\rangle$	B1	$\{(0,1),(1,2),(2,0)\}$	$\{(0,0),(1,1),(2,3),(3,2),(4,4),(5,5)\}$
$\langle B4, m2\rangle$	B1	$\{(0,1),(1,2),(2,4)\}$	$\{(0,4),(1,1),(2,3),(3,2),(4,0),(5,5)\}$



Rysunek 5.2: Zbiór $\mathcal{B}$ dla przykładu 5.1 - wymagającego współpracy pomiędzy dronami wraz z zaznaczoną relacją reakcji pomiędzy dwugrafami. Dwukierunkowe strzałki przy dowolnej parze dwugrafów Bx, By oznaczają, że zarówno $Bx \xrightarrow{r} By$ należy do relacji reakcji, jak i $By \xrightarrow{r} Bx$.

Tabela 5.3: Funkcje postępu misji dla przykładu 5.1. Kolejne funkcje odpowiadają kolejnym przejściom w TTS, zdefiniowanym w tabeli 5.2.

Dla skrócenia zapisu wprowadzono następującą notację.
Zapis skrócony: $f_x(c,t) \to [\langle (b, y+2), (a, x)\rangle \pm r_{t+1}]$
jest równoznaczny funkcji postępu misji postaci:

$$f_x(c,t) = \begin{cases} [\langle (b, y+2), (a, x)\rangle, \mathbb{H}' \cup \{r_{t+1}\}] & c = [\langle (a, x), (b, y)\rangle, \mathbb{H}'] \\ 0 & c = 0 \end{cases}$$

$f_1(c,t) \to \left[\langle (a, x+1), (b, y)\rangle \pm m1^1_{t+1}\right]$	$f_2(c,t) \to \left[\langle (b, y+3), (a, x)\rangle \pm m2^2_{t+1}\right]$
$f_3(c,t) \to \left[\langle (a, x+1), (b, y)\rangle \pm m1^3_{t+1}\right]$	$f_4(c,t) \to \left[\langle (b, y+3), (a, x)\rangle \pm m2^4_{t+1}\right]$
$f_5(c,t) \to \left[\langle (a, x+1), (b, y)\rangle \pm m1^5_{t+1}\right]$	$f_6(c,t) \to \left[\langle (b, y), (a, x+3)\rangle \pm m2^6_{t+1}\right]$
$f_7(c,t) \to \left[\langle (a, x), (b, y+1)\rangle \pm m1^7_{t+1}\right]$	$f_8(c,t) \to \left[\langle (b, y), (a, x+3)\rangle \pm m2^8_{t+1}\right]$
$f_9(c,t) \to \left[\langle (b, y+3), (a, x)\rangle \pm m2^9_{t+1}\right]$	$f_{10}(c,t) \to \left[\langle (a, x+1), (b, y)\rangle \pm m1^{10}_{t+1}\right]$
$f_{11}(c,t) \to \left[\langle (b, z+4), (a, z+4)\rangle \pm m3^{11}_{t+1}\right]$	$f_{12}(c,t) \to \left[\langle (a, x), (b, y+1)\rangle \pm m1^{12}_{t+1}\right]$
$f_{13}(c,t) \to \left[\langle (a, x), (b, y+1)\rangle \pm m1^{13}_{t+1}\right]$	$f_{14}(c,t) \to \left[\langle (b, y), (a, x+3)\rangle \pm m2^{14}_{t+1}\right]$
$f_{15}(c,t) \to \left[\langle (b, y), (a, x+3)\rangle \pm m2^{15}_{t+1}\right]$	

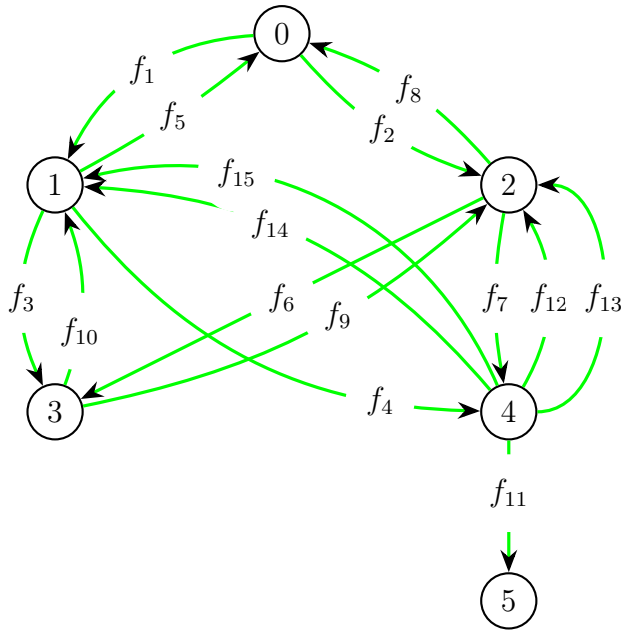
Pierwsza macierz ma postać:

$$\mathbb{M}^t_F = \begin{bmatrix} f_{null} & f_1 & f_2 & f_{null} & f_{null} & f_{null} \\ f_5 & f_{null} & f_{null} & f_3 & f_4 & f_{null} \\ f_8 & f_{null} & f_{null} & f_6 & f_7 & f_{null} \\ f_{null} & f_{10} & f_9 & f_{null} & f_{null} & f_{null} \\ f_{null} & f_{14} + f_{15} & f_{12} + f_{13} & f_{null} & f_{null} & f_{11} \\ f_{null} & f_{null} & f_{null} & f_{null} & f_{null} & f_{null} \end{bmatrix}$$

Wiedząc, że stan początkowy zadania jest reprezentowany przez wierzchołek 0 w przestrzeni stanów oraz że w systemie znajdują się dwaj agenci, możemy zapisać drugą ze wspomnianych macierzy jako:

$$\mathbb{M}^0_K = \begin{bmatrix} [\langle (1,0), (2,0)\rangle, \emptyset] & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Przechodząc teraz do wyznaczenia polityki zachowania szukane są marszruty, prowadzące do stanu utożsamianego z wykonaniem zadania, czyli do stanu systemu reprezentowanego przez wierzchołek 5 na graficznej postaci *SS*. Pierwsze takie marszruty będą miały długość 5 kroków. Oznacza to, że będą znajdowały się w 6 kolumnie (licząc od 0) macierzy $\mathbb{M}^5_K$. Poniżej wypisano wszystkie elementy szeregu w tej kolumnie:



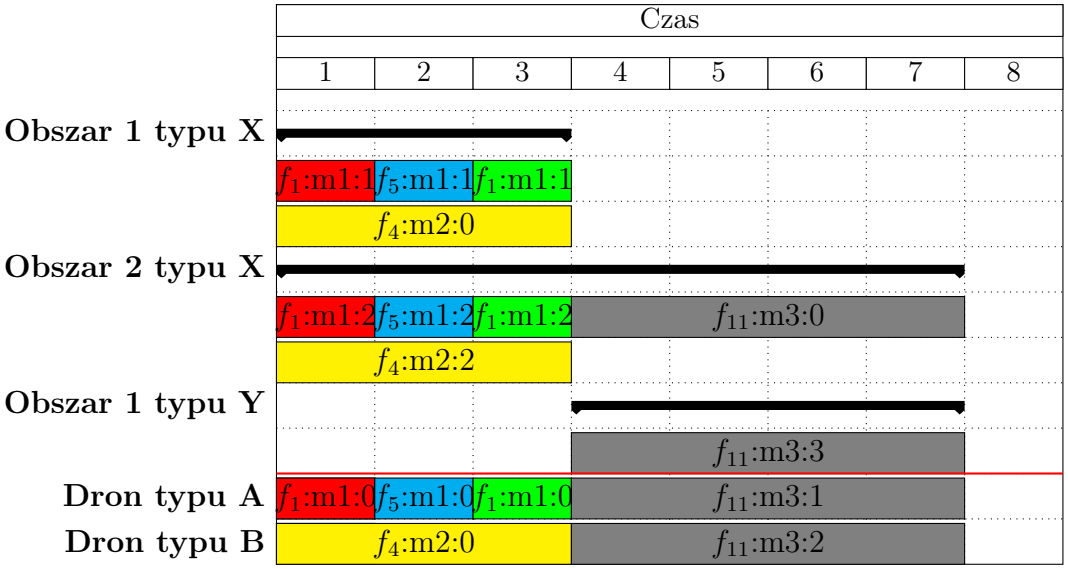
Rysunek 5.3: Przestrzeń stanów dla przykładu 5.1 - zadania wymagającego współpracy między dronami.

- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m2_4^4, m1_3^1, m1_2^5, m1_1^1\}]$,
- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m2_4^4, m1_3^1, m1_2^3, m1_1^1\}]$,
- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m2_3^2, m1_2^5, m1_1^1\}]$,
- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m2_3^9, m1_2^3, m1_1^1\}]$,
- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m2_4^7, m1_3^{13}, m2_2^4, m1_1^1\}]$,
- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m1_3^{13}, m1_2^7, m2_1^2\}]$,
- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m1_3^{12}, m2_2^4, m1_1^1\}]$,
- $[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m1_3^{12}, m1_2^7, m2_1^2\}]$.

Szczegółowy przebieg procesu konstrukcji harmonogramu działań dla elementów zadania na podstawie polityki zachowania będzie przedmiotem następnego rozdziału. Ten przykład ograniczony zostanie do przedstawienia samego wyniku w postaci graficznej. Harmonogram działań skonstruowany w oparciu o marszrutę:

$$0 \xrightarrow{f_1} 1 \xrightarrow{f_5} 0 \xrightarrow{f_1} 1 \xrightarrow{f_4} 4 \xrightarrow{f_{11}} 5$$

będzie miał postać jak na rysunku 5.4.



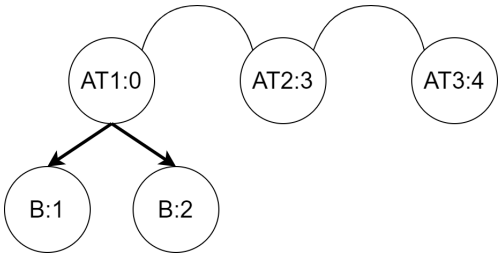
Rysunek 5.4: Harmonogram działań dla przykładu zadania wymagającego współpracy pomiędzy dronami, opisanego w rozdziale 5.1. Został on skonstruowany na podstawie marszruty $0 \xrightarrow{f_1} 1 \xrightarrow{f_5} 0 \xrightarrow{f_1} 1 \xrightarrow{f_4} 4 \xrightarrow{f_{11}} 5$. Nie uwzględniono na nim trzeciego obszaru typu X, ponieważ nie uczestniczy on w żadnej czynności. Czynności wykonywane przez obydwa drony w tych samych przedziałach czasu, oznaczone takim samym kolorem są działaniami kooperacyjnymi.

5.2 Weryfikacja modelu

Kolejny przykład ma na celu:

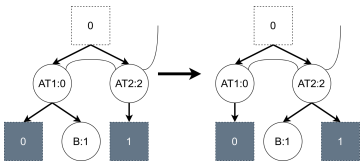
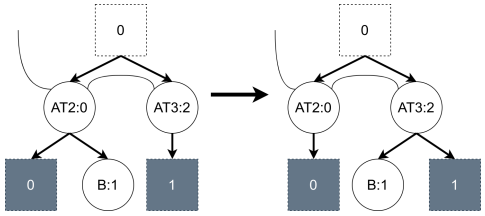
- przedstawienie w jaki sposób, wykorzystując opracowaną wcześniej marszrutę w przestrzeni stanów, można zweryfikować model;
- skonstruować kolejne stany scenariusza;
- przekształcić wejściową marszrutę w harmonogram działań.

Zadanie w przykładzie 5.2 polega na przemieszczeniu dwóch dronów pomiędzy trzema obszarami. Tym razem drony są tego samego typu (oznaczanym przez B), zaś wszystkie obszary różnią się od siebie. Kolejne typy obszarów oznaczymy przez *AT1* , *AT2* i *AT3*. Są one połączone liniowo tzn. z obszaru *AT1* można przejść do *AT2*, a stamtąd do *AT3*. Na każdy obszar można wejść, ale nie można z niego wyjść. Obydwa drony na początku misji znajdują się na obszarze *AT1* i ich celem jest przemieszczenie się do *AT3*. Każdy dron porusza się samodzielnie i zajmuje mu to 1 jednostkę czasu.



Rysunek 5.5: Stan początkowy dla przykładu weryfikacji poprawnego modelu.

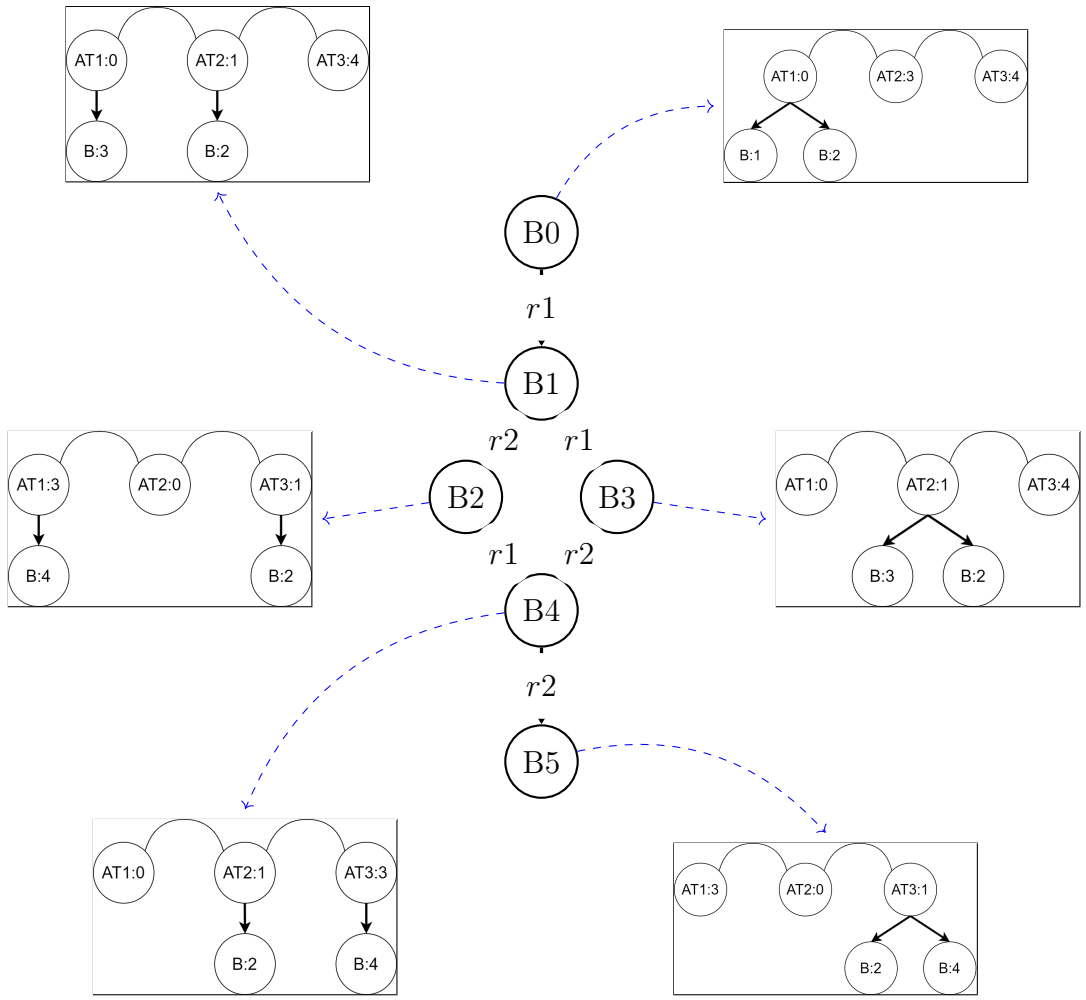
Tabela 5.4: Elementy zbioru $\mathcal{R}$ TBRS dla przykładu weryfikacji poprawności modelu. Funkcje τ są tożsamościami.

Nazwa reguły reakcji	Reprezentacja graficzna
$r1$	
$r2$	

Na podstawie powyższej definicji zadania można zidentyfikować dwie reguły reakcji, przedstawione w tabeli 5.4. Stan początkowy zadania w postaci dwugrafu został zobrazowany na rysunku 5.5.

Dla stanu początkowego z rysunku 5.5 oraz zbioru reguł reakcji z tabeli 5.4 wygenerowano dwugrafowy system reakcyjny, przedstawiony wraz ze zbiorem *Agt* na rysunku 5.6. Ten TBRS można przekształcić do śledzącego systemu przejść, przedstawionego w tabeli 5.5. Ostatnim elementem modelu jest przekształcenie wspomnianego TTS w przestrzeń stanów. Jej postać graficzna nie zostanie tutaj zamieszczona, ale kolejne stany odpowiadają numeracji kolejnych dwugrafów z TBRS. Funkcje postępu misji (będące wynikiem przekształcenia kolejnych przejść z TTS) zostały zamieszczone w tabeli 5.6.

Polityką zachowania, w oparciu o którą będzie weryfikowany model jest $0 \xrightarrow{f_1} 1 \xrightarrow{f_3} 2 \xrightarrow{f_5} 4 \xrightarrow{f_8} 5$.



Rysunek 5.6: Zbiór *Agt* dla TBRS z przykładu weryfikacji poprawności modelu.

Tabela 5.5: Śledzący system przejść dla przykładu weryfikacji modelu. Każdy wiersz definiuje pojedyncze przejście.

Apl	Agt	Par	Res
$\langle B0, r1\rangle$	$B1$	$\{(0,0),(1,2),(2,3)\}$	$\{(0,0),(1,3),(2,2),(3,1),(4,4)\}$
$\langle B0, r1\rangle$	$B1$	$\{(0,0),(1,1),(2,3)\}$	$\{(0,0),(1,3),(2,1),(3,2),(4,4)\}$
$\langle B1, r2\rangle$	$B2$	$\{(0,1),(1,2),(2,4)\}$	$\{(0,1),(1,4),(2,2),(3,0),(4,3)\}$
$\langle B1, r1\rangle$	$B3$	$\{(0,0),(1,3),(2,1)\}$	$\{(0,0),(1,1),(2,2),(3,3),(4,4)\}$
$\langle B2, r1\rangle$	$B4$	$\{(0,3),(1,4),(2,0)\}$	$\{(0,3),(1,0),(2,4),(3,1),(4,2)\}$
$\langle B3, r2\rangle$	$B4$	$\{(0,1),(1,2),(2,4)\}$	$\{(0,0),(1,1),(2,3),(3,4),(4,2)\}$
$\langle B3, r2\rangle$	$B4$	$\{(0,1),(1,3),(2,4)\}$	$\{(0,0),(1,1),(2,2),(3,4),(4,3)\}$
$\langle B4, r2\rangle$	$B5$	$\{(0,1),(1,2),(2,3)\}$	$\{(0,1),(1,3),(2,4),(3,0),(4,2)\}$

Tabela 5.6: Funkcje postępu misji dla przykładu weryfikacji modelu. Do zapisu funkcji został użyty skrócony zapis zdefiniowany w tabeli 5.3. Kolejne funkcje odpowiadają kolejnym przejściom z TTS zdefiniowanego w tabeli 5.5.

$f_1(c,t) \rightarrow \left[\langle (b,y+1),(a,x)\rangle \pm r1^1_{t+1}\right]$	$f_2(c,t) \rightarrow \left[\langle (a,x+1),(b,y)\rangle \pm r1^2_{t+1}\right]$
$f_3(c,t) \rightarrow \left[\langle (a,x+1),(b,y)\rangle \pm r2^3_{t+1}\right]$	$f_4(c,t) \rightarrow \left[\langle (a,x),(b,y+1)\rangle \pm r1^4_{t+1}\right]$
$f_5(c,t) \rightarrow \left[\langle (b,y+1),(a,x)\rangle \pm r1^5_{t+1}\right]$	$f_6(c,t) \rightarrow \left[\langle (b,y),(a,x+1)\rangle \pm r2^6_{t+1}\right]$
$f_7(c,t) \rightarrow \left[\langle (a,x),(b,y+1)\rangle \pm r2^7_{t+1}\right]$	$f_8(c,t) \rightarrow \left[\langle (b,y),(a,x+1)\rangle \pm r2^8_{t+1}\right]$

Mając opracowany model roju i politykę zachowania można przystąpić do weryfikacji modelu i opracowania harmonogramu działań dla wszystkich elementów zadania.

Danymi wejściowymi dla algorytmu jest marszruta. Marszruta wejściowa w postaci zbioru ma postać $W = \{(1, f_1), (2, f_3), (3, f_5), (4, f_8)\}$.

Ze względu na obszerność przebiegu algorytmu zaprezentowanego w tabeli 5.7 nie został tam uwzględniony przebieg etapu 2. Szczegółowy przebieg procedury rozszerzania marszruty dla tego przykładu został zamieszczony w załączniku F.

Rozszerzona marszruta ma postać:

$$W_M = \{e_1, e_2, e_3, e_4\}$$

przy czym, jej elementy są następujące:

- $e_1 = (1, f_1, \{(0,0),(2,1),(3,2)\}, \{(0,0),(1,1),(2,2),(3,3),(4,4)\}, 5, (\{2\}, 1))$,
- $e_2 = (2, f_3, \{(3,0),(2,1),(4,2)\}, \{(0,0),(3,1),(2,2),(1,3),(4,4)\}, 5, (\{2\}, 1))$,
- $e_3 = (3, f_5, \{(0,0),(1,1),(3,2)\}, \{(0,3),(1,4),(2,2),(3,0),(4,1)\}, 5, (\{1\}, 1))$,
- $e_4 = (4, f_8, \{(3,0),(1,1),(4,2)\}, \{(0,0),(1,2),(2,4),(3,1),(4,3)\}, 5, (\{1\}, 1))$.

Tabela 5.7: Kolejne kroki algorytmu weryfikacji poprawności modelu MAS, dla przykładu mającego na celu demonstrację przebiegu weryfikacji poprawnego modelu.

Etap	**Krok**	**Wynik/Komentarz**
1	$S_r \uplus \{(d-1, s, m_s, i_s)\}$	$S_r = \left\{ \begin{pmatrix} 0, \text{B0}, \\ \{(0,0),(1,1),(2,2),(3,3),(4,4)\}, \\ ((1,0),(2,0)) \end{pmatrix} \right\}$
1	$\rightarrow Etap3(\dots)$	
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 1$ $t_f = f_1$ $m_r = \{(0,0),(2,1),(3,2)\}$ $m_{full} = \{(0,0),(1,1),(2,2),(3,3),(4,4)\}$ $n_x = 5$ $A_d = \{2\}$ $n_d = 1$ $W_c = W_c \setminus \{e_1\} = \{e_2, e_3, e_4\}$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,0),(2,1))$
3	$A_f = Objects_F(i_n, d)$	$A_f = \emptyset$
3	$t_{tra} = Corr_{Tra}(t_f)$	t_{tra} = pierwszy wiersz tabeli 5.5
3	$i_c = i_n$	$i_c = ((1,0),(2,1))$
3	$r = Corr_{\mathcal{R}}(t_{tra})$	r = reguła reakcji *r1*
3	$result = Etap4(\dots)$	Etap 4 zakończony bez błędu
3	$s_c, m_c, n_x = result$	$s_c = \text{B1}$ $m_c = \{(0,0),(3,1),(2,2),(1,3),(4,4)\}$ (m_c jest wyliczana tak samo jak m'_{full} w etapie 2) $n_x = 5$
Kontynuacja na następnej stronie		

Tabela 5.7 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
3	$plan^* \cup Plan(f, m_r, d, n_d)$	$plan^* = \left(\begin{array}{r} Assignment = \left\{ \begin{array}{l} (0, (f_1, 0)), \\ (2, (f_1, 0)), \\ (3, (f_1, 0)) \end{array} \right\} \\ role = \left\{ \begin{array}{l} ((0, (f_1, 0)), 0), \\ ((2, (f_1, 0)), 1), \\ ((3, (f_1, 0)), 2) \end{array} \right\} \\ duration = \{(f_1, 1)\} \end{array} \right)$
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 2$ $t_f = f_3$ $m_r = \{(3,0), (2,1), (4,2)\}$ $m_{full} = \{(0,0), (3,1), (2,2), (1,3), (4,4)\}$ $n_x = 5$ $A_d = \{2\}$ $n_d = 1$ $W_c = W_c \setminus \{e_2\} = \{e_3, e_4\}$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,0), (2,2))$
3	$A_f = Objects_F(i_n, d)$	$A_f = \{2\}$
3	$A_o = A_o \cup A_f$	$A_o = \emptyset \cup \{2\} = \{2\}$
3	$W_o \uplus \{(n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d))\}$	$W_o = \emptyset \cup \{e_2\}$
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 3$ $t_f = f_5$ $m_r = \{(0,0), (1,1), (3,2)\}$ $m_{full} = \{(0,3), (1,4), (2,2), (3,0), (4,1)\}$ $n_x = 5$ $A_d = \{1\}$ $n_d = 1$ $W_c = W_c \setminus \{e_3\} = \{e_4\}$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,1), (2,1))$
Kontynuacja na następnej stronie		

Tabela 5.7 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
3	$A_f = Objects_F(i_n, d)$	$A_f = \emptyset$
3	$t_{tra} = Corr_{Tra}(t_f)$	t_{tra} = piąty wiersz tabeli 5.5
3	$i_c = i_n$	$i_c = ((1,1),(2,1))$
3	$r = Corr_{\mathcal{R}}(t_{tra})$	r = reguła reakcji *r1*
3	$result = Etap4(\dots)$	Etap 4 zakończony bez błędu
3	$s_c, m_c, n_x = result$	$s_c = \text{B3}$ $m_c = \{(0,0),(3,1),(2,2),(1,3),(4,4)\}$ (Użyto czwartego przejścia z tabeli 5.5 w trakcie etapu 4) $n_x = 5$
3	$plan^* \cup Plan(f, m_r, d, n_d)$	$plan^* \cup \left(\begin{array}{r} Assignment = \left\{ \begin{array}{l} (0,(f_5,0)), \\ (1,(f_5,0)), \\ (3,(f_5,0)) \end{array} \right. \\ role = \left\{ \begin{array}{l} ((0,(f_5,0)),0), \\ ((1,(f_5,0)),1), \\ ((3,(f_5,0)),2) \end{array} \right. \\ duration = \{(f_5,1)\} \end{array} \right)$
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 4$ $t_f = f_8$ $m_r = \{(3,0),(1,1),(4,2)\}$ $m_{full} = \{(0,0),(1,2),(2,4),(3,1),(4,3)\}$ $n_x = 5$ $A_d = \{1\}$ $n_d = 1$ $W_c = W_c \setminus \{e_4\} = \emptyset$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,2),(2,1))$
3	$A_f = Objects_F(i_n, d)$	$A_f = \{1\}$
3	$A_o = A_o \cup A_f$	$A_o = \{2\} \cup \{1\} = \{1,2\}$
		Kontynuacja na następnej stronie

Tabela 5.7 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
3	$W_o \uplus \{(n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d))\}$	$W_o = \{e_2\} \cup \{e_4\}$
3	Koniec - ok	
1	$result \leftarrow Etap3(\dots)$	Etap 3 zakończony
1	$W_c, s, m_s, i_s = result$	$W_c = \{e_2, e_4\}$ $s = \text{B3}$ $m_s = \{(0,0), (3,1), (2,2), (1,3), (4,4)\}$ $i_s = ((1,1), (2,1))$
1	$d = d + 1$	$d = 2$
1	$plan = plan \cup plan^*$	$plan = plan^*$
1	$S_r \uplus \{(d-1, s, m_s, i_s)\}$	$S_r \uplus \left\{ \begin{pmatrix} 1, \text{B3}, \\ \{(0,0), (3,1), (2,2), (1,3), (4,4)\}, \\ ((1,1), (2,1)) \end{pmatrix} \right\}$
1	$\rightarrow Etap3(\dots)$	Rozpoczęcie etapu 3
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 2$ $t_f = f_3$ $m_r = \{(3,0), (2,1), (4,2)\}$ $m_{full} = \{(0,0), (3,1), (2,2), (1,3), (4,4)\}$ $n_x = 5$ $A_d = \{2\}$ $n_d = 1$ $W_c = W_c \setminus \{e_2\} = \{e_4\}$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,1), (2,2))$
3	$A_f = Objects_F(i_n, d)$	$A_f = \emptyset$
3	$t_{tra} = Corr_{Tra}(t_f)$	t_{tra} = trzeci wiersz tabeli 5.5
3	$i_c = i_n$	$i_c = ((1,1), (2,2))$
3	$r = Corr_{\mathcal{R}}(t_{tra})$	r = reguła reakcji *r2*
3	$result = Etap4(\dots)$	Etap 4 zakończony bez błędu
Kontynuacja na następnej stronie		

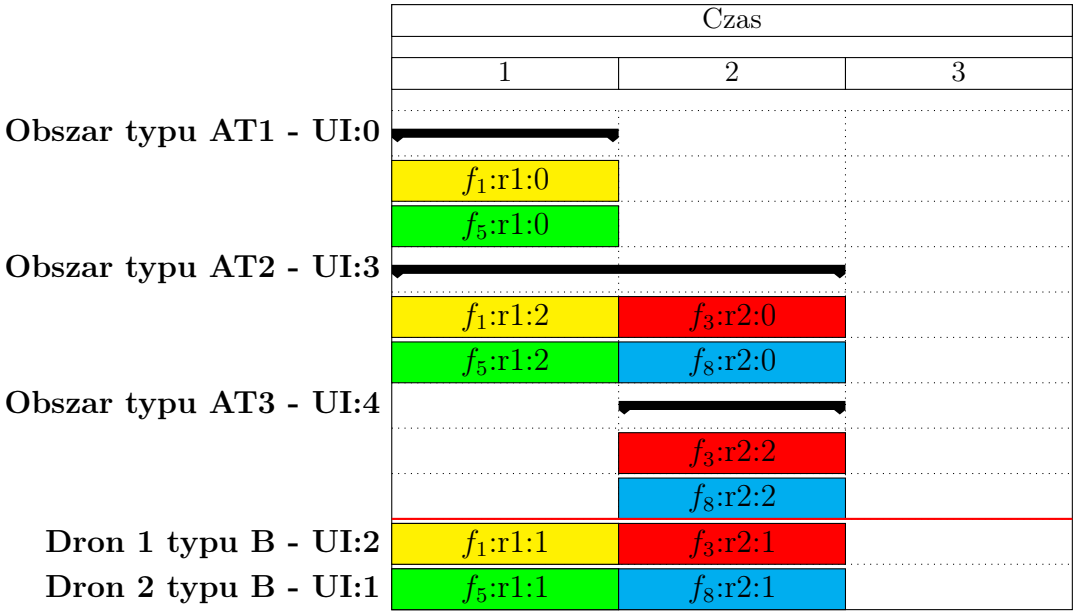
Tabela 5.7 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
3	$s_c, m_c, n_x = result$	$s_c = \text{B4}$ $m_c = \{(0,0),(3,1),(4,3),(1,2),(2,4)\}$ (Użyto szóstego przejścia z tabeli 5.5 w trakcie etapu 4) $n_x = 5$
3	$plan^* \cup Plan(f, m_r, d, n_d)$	$plan^* \cup \left(\begin{array}{r} Assignment = \left\{ \begin{array}{l} (3,(f_3,1)), \\ (2,(f_3,1)), \\ (4,(f_3,1)) \end{array} \right. \\ role = \left\{ \begin{array}{l} ((3,(f_3,1)),0), \\ ((2,(f_3,1)),1), \\ ((4,(f_3,1)),2) \end{array} \right. \\ duration = \{(f_3,1)\} \end{array} \right)$
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 4$ $t_f = f_8$ $m_r = \{(3,0),(1,1),(4,2)\}$ $m_{full} = \{(0,0),(1,2),(2,4),(3,1),(4,3)\}$ $n_x = 5$ $A_d = \{1\}$ $n_d = 1$ $W_c = W_c \setminus \{e_4\} = \emptyset$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,2),(2,2))$
3	$A_f = Objects_F(i_n, d)$	$A_f = \emptyset$
3	$t_{tra} = Corr_{Tra}(t_f)$	$t_{tra} =$ ósmy wiersz tabeli 5.5
3	$i_c = i_n$	$i_c = ((1,2),(2,2))$
3	$r = Corr_{\mathcal{R}}(t_{tra})$	$r =$ reguła reakcji *r2*
3	$result = Etap4(\dots)$	Etap 4 zakończony bez błędu
Kontynuacja na następnej stronie		

Tabela 5.7 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
3	$s_c, m_c, n_x = result$	$s_c = \text{B5}$ $m_c = \{(0,3),(3,0),(4,1),(1,4),(2,2)\}$ $n_x = 5$
3	$plan^* \cup Plan(f, m_r, d, n_d)$	$plan^* \cup \left(\begin{array}{r} Assignment = \left\{ \begin{array}{l} (3,(f_8,1)), \\ (1,(f_8,1)), \\ (4,(f_8,1)) \end{array} \right. \\ role = \left\{ \begin{array}{l} ((3,(f_8,1)),0), \\ ((1,(f_8,1)),1), \\ ((4,(f_8,1)),2) \end{array} \right. \\ duration = \{(f_8,1)\} \end{array} \right)$
3	Koniec - ok	
1	$result \leftarrow Etap3(\dots)$	Etap 3 zakończony
1	$W_c, s, m_s, i_s = result$	$W_c = \emptyset$ $s = \text{B5}$ $m_s = \{(0,3),(3,0),(4,1),(1,4),(2,2)\}$ $i_s = ((1,2),(2,2))$
1	$d = d + 1$	$d = 3$
1	$plan = plan \cup plan^*$	Harmonogram zostaje zaktualizowany o elementy zbiorów składowych $plan^*$.
1	$S_r \uplus \{(d-1, s, m_s, i_s)\}$	$S_r \uplus \left\{ \left(\begin{array}{r} 2, \text{B5}, \\ \{(0,3),(3,0),(4,1),(1,4),(2,2)\}, \\ ((1,2),(2,2)) \end{array} \right) \right\}$
1	Koniec - ok	

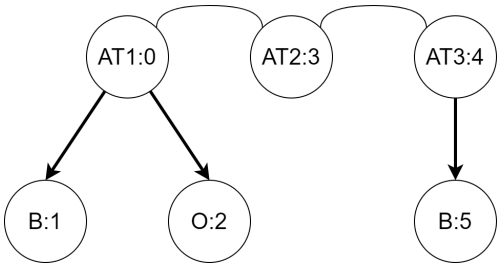
Jeden ze stanów scenariusza będący wynikiem weryfikacji przedstawia sytuację, w której obydwa drony znajdują się na obszarze *AT2* pomimo tego, że marszruta wejściowa nie „przechodziła" przez wierzchołek, oznaczający taki stan sytemu. Można to sprawdzić śledząc kolejne stany (odpowiadające kolejnym dwugrafom ze zbioru Agt), przez które przechodzi marszruta $0 \xrightarrow{f_1} 1 \xrightarrow{f_3} 2 \xrightarrow{f_5} 4 \xrightarrow{f_8} 5$.



Rysunek 5.7: Harmonogram działań elementów zadania z przykładu 5.2. Po lewej stronie diagramu zamieszczono opis elementu zadania wraz z jego unikalnym identyfikatorem.

Harmonogram działań elementów zadania dla tego scenariusza przedstawiono na rysunku 5.7. Obydwa drony wykonują czynności *r1* i *r2* równolegle (niezależnie od siebie). W odróżnieniu od poprzednio przedstawionych harmonogramów, ten uwzględnia unikalne identyfikatory przypisane do elementów zadania występujących w scenariuszu.

Funkcje powiązane z każdym stanem scenariusza pozwalają śledzić unikalne obiekty. Dla przykładu, funkcja $m_s = (0,0),(1,1),(2,2),(3,3),(4,4)$ dla stanu w momencie 0 wskazuje, że obiekt o unikalnym identyfikatorze 2 (argument funkcji m_s) jest reprezentowany przez wierzchołek o identyfikatorze 2 (wartość funkcji m_s dla argumentu 2). Sama podstawa dwugrafu jest niewystarczająca do śledzenia obiektów, co można zobaczyć porównując stany scenariusza w kolejnych momentach czasu. Dla stanu w momencie 1 UI o wartości 3 wskazuje na wierzchołek o identyfikatorze 1. Oznacza to, że jest to ten sam element zadania, który w momencie 0 jest reprezentowany przez wierzchołek o identyfikatorze 3 oraz ten sam element, który w momencie 2 reprezentowany jest przez wierzchołek o identyfikatorze 0.



Rysunek 5.8: Stan początkowy dla przykładu weryfikacji niepoprawnego modelu.

5.3 Wykrycie błędów w modelu

Kolejny przykład ma na celu zademonstrowanie, w jaki sposób opracowany algorytm weryfikacji modeli MAS wykrywa w nich błędy. Przedstawiony zostanie nieprawidłowy model systemu wieloagentowego wraz z uzasadnieniem, co jest tego przyczyną oraz jak może on zostać poprawiony.

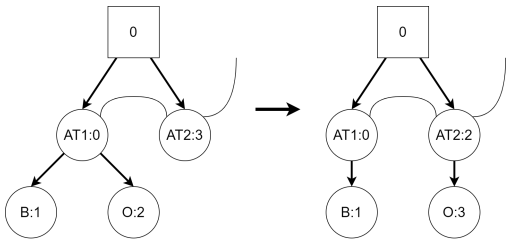
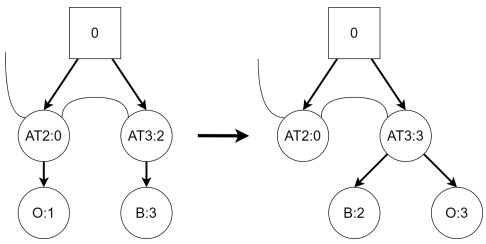
Zadanie na potrzeby przykładu 5.3 składa się z dwóch dronów, których celem jest przeniesienie pewnego obiektu. Obydwa drony są oddzielone obszarem pośredniczącym, na który musi zostać przeniesiony obiekt w celu przetransportowania. Obszary tak jak w poprzednim przykładzie nazwiemy *AT1*,*AT2* oraz *AT3*. Obszary są ze sobą połączone w sposób identyczny, jak to miało miejsce w poprzednim przykładzie, czyli AT1 z AT2, a AT2 zarówno z AT1 jak i AT3. Obydwa użyte drony są tego samego typu (oznaczanym przez B), który jest w stanie przemieścić obiekt (oznaczony przez O) pomiędzy dwoma obszarami AT1 i AT2 oraz z AT2 do AT3. W początkowym stanie systemu drony zajmują obszary AT1 i AT3, a obiekt przeznaczony do przesunięcia znajduje się na obszarze AT1.

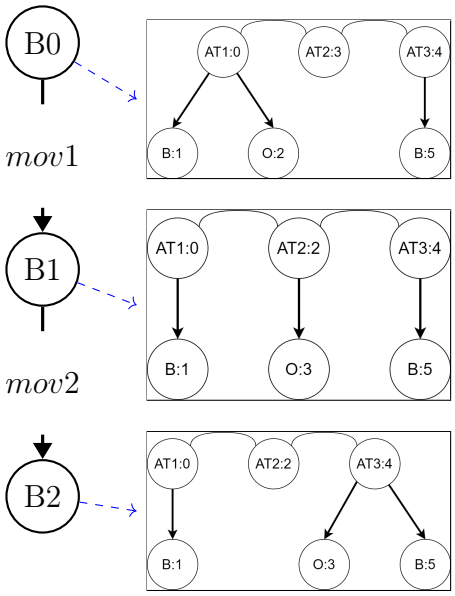
W oparciu o powyższy opis można zidentyfikować dwie reguły reakcji przedstawione w tabeli 5.8. Graficzna reprezentacja stanu początkowego została zamieszczona na rysunku 5.8, a skonstruowany na jej podstawie (i wspomnianych reguł reakcji) TBRS przedstawiono na rysunku 5.9. Ten z kolei może zostać przekształcony w TTS z tabeli 5.9.

Kolejnym krokiem jest przekształcenie śledzącego systemu przejść w przestrzeń stanów. Należy w tym celu pogrupować wszystkie zidentyfikowane elementy zadania. Na potrzeby tego przykładu dokonamy następującej kategoryzacji wcześniej zidentyfikowanych elementów:

1. elementy otoczenia - $\{AT1, AT2, AT3, O\}$,

Tabela 5.8: Elementy zbioru $\mathcal{R}$ TBRS dla przykładu weryfikacji nieprawidłowego modelu.

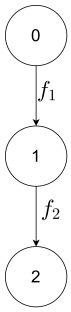
Nazwa reguły reakcji	Reprezentacja graficzna
$mov1$	
$mov2$	



Rysunek 5.9: Zbiór *Agt* dla TBRS z przykładu weryfikacji nieprawidłowego modelu.

Tabela 5.9: Śledzący system przejść dla przykładu weryfikacji nieprawidłowego modelu. Każdy wiersz definiuje pojedyncze przejście.

Apl	Agt	Par	Res
$\langle B0, mov1\rangle$	$B1$	$\{(0,0),(1,1),(2,2),(3,3)\}$	$\{(0,0),(1,1),(2,3),(3,2),(4,4),(5,5)\}$
$\langle B1, mov2\rangle$	$B2$	$\{(0,2),(1,3),(2,4),(3,5)\}$	$\{(0,0),(1,1),(2,2),(3,3),(4,4),(5,5)\}$



Rysunek 5.10: Przestrzeń stanów skonstruowana na podstawie TTS z tabeli 5.9 oraz kategoryzacji elementów zadania, w której tylko drony są zakwalifikowane do obiektów. Funkcje postępu misji zostały zdefiniowane w tabeli 5.10.

Tabela 5.10: Funkcje postępu misji dla przykładu weryfikacji nieprawidłowego modelu. Do zapisu funkcji został użyty skrócony zapis zdefiniowany w tabeli 5.3. Kolejne funkcje odpowiadają kolejnym przejściom z TTS zdefiniowanego w tabeli 5.9.

F.P.M	Funkcja postępu misji w formie skróconej
f_1	$f_1(c,t) \rightarrow [\langle (a, x+1), (b, y) \rangle \pm mov1_{t+1}]$
f_2	$f_2(c,t) \rightarrow [\langle (a, x), (b, y+1) \rangle \pm mov2_{t+1}]$

2. obiekty bierne - $\emptyset$,

3. obiekty czynne - $\{B\}$.

Należy tutaj podkreślić, że jest to błędna kategoryzacja, ale w tym przykładzie jest to celowe. Wyjaśnienie dlaczego powyższy podział jest błędny nastąpi po przedstawieniu przebiegu algorytmu weryfikacji.

Śledzący system przejść z tabeli 5.9 uwzględniający powyższy podział elementów zadania oraz przyjmując, że obydwie czynności zajmą 1 jednostkę czasu można przekształcić w przestrzeń stanów jak na rysunku 5.10.

Jedyna marszruta jakiej możemy użyć do konstrukcji polityki zachowania to $0 \xrightarrow{f_1} 1 \xrightarrow{f_2} 2$. Można ją przedstawić w postaci zbioru $W = \{(1, f_1), (2, f_2)\}$.

Tak samo jak w poprzednim przykładzie, w opisie przebiegu algorytmu weryfikacji zostanie pominięty przebieg etapu 2. Wynik rozszerzenia powyższej marszruty w 2 etapie jest następujący:

$$W_M = \{e_1, e_2\}$$

gdzie kolejne jej elementy mają postać:

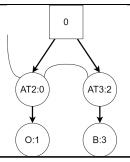
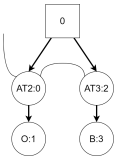
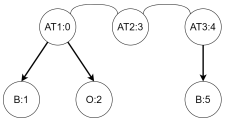
1. $(1, f_1, \{(0,0),(1,1),(2,2),(3,3)\}, \{(0,0),(1,1),(2,2),(3,3),(4,4),(5,5)\}, 6, (\{1\}, 1))$,

2. $(2, f_2, \{(2,0),(3,1),(4,2),(5,3)\}, \{(0,0),(1,1),(2,3),(3,2),(4,4),(5,5)\}, 6, (\{2\}, 1))$.

Tabela 5.11: Przebieg weryfikacji przykładowego nieprawidłowego modelu.

Etap	**Krok**	**Wynik/Komentarz**
1	$S_r \uplus \{((d-1, s, m_s, i_s))\}$	$S_r = \left\{ \left(\begin{array}{r} 0, \mathrm{B0}, \\ \{(0,0),(1,1),(2,2),(3,3),(4,4),(5,5)\}, \\ ((1,0),(2,0)) \end{array} \right) \right\}$
1	$\rightarrow Etap3(\dots)$	
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 1$ $t_f = f_1$ $m_r = \{(0,0),(1,1),(2,2),(3,3)\}$ $m_{full} = \Big\{(0,0),(1,1),(2,2),(3,3),(4,4),(5,5)\Big\}$ $n_x = 6$ $A_d = \{1\}$ $n_d = 1$ $W_c = W_c \setminus \{e_1\} = \{e_2\}$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,1),(2,0))$
3	$A_f = Objects_F(i_n, d)$	$A_f = \emptyset$
3	$t_{tra} = Corr_{Tra}(t_f)$	t_{tra} = pierwszy wiersz tabeli 5.9
3	$i_c = i_n$	$i_c = ((1,1),(2,0))$
3	$r = Corr_{\mathcal{R}}(t_{tra})$	r = reguła reakcji *mov1*
3	$result = Etap4(\dots)$	Etap 4 zakończony bez błędu
3	$s_c, m_c, n_x = result$	$s_c = \mathrm{B1}$ $m_c = \{(0,0),(1,1),(2,3),(3,2),(4,4),(5,5)\}$ $n_x = 6$
3	$plan^* \cup Plan(\dots)$	Wynik nieistotny w tym przykładzie.
Kontynuacja na następnej stronie		

Tabela 5.11 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
3	$n_f, t_f, m_r, m_{full}, n_x, (A_d, n_d), W_c = First_M(W_c)$	$n_f = 2$ $t_f = f_2$ $m_r = \{(2,0),(3,1),(4,2),(5,3)\}$ $m_{full} = \Big\{(0,0),(1,1),(2,3),(3,2),(4,4),(5,5)\Big\}$ $n_x = 6$ $A_d = \{2\}$ $n_d = 1$ $W_c = W_c \setminus \{e_2\} = \emptyset$
3	$i_n = Objects_U(i_c, (A_d, n_d))$	$i_n = ((1,1),(2,1))$
3	$A_f = Objects_F(i_n, d)$	$A_f = \emptyset$
3	$t_{tra} = Corr_{Tra}(t_f)$	t_{tra} = drugi wiersz tabeli 5.9
3	$i_c = i_n$	$i_c = ((1,1),(2,1))$
3	$r = Corr_{\mathcal{R}}(t_{tra})$	r = reguła reakcji *mov2*
3	$\rightarrow Etap4(\dots)$	
4	$r_l = Corr_{Red}(r)$	$r_l =$ 
4	$c_1 = IsUpdatePossible(\dots)$	$c_1 = false$ Wzorzec:  nie występuje w dwugrafie: 
1	Koniec - błąd	

Skutkiem tego, że przedstawiony model jest błędnie wykonany jest to, że redeks reguły *mov2* nie występuje w stanie systemu utożsamianym z dwugrafem B0. Niepoprawność modelu wynika z tego, że przenoszony obiekt został zakwalifikowany jako element otoczenia, a nie jako obiekt bierny (w rozumieniu terminologii zdefiniowanej na początku rozdziału 3). Z tego powodu konstruując politykę za-

chowania nie jest brany pod uwagę upływ czasu dla obiektu przenoszonego. To z kolei powoduje, że w ramach procesu wyznaczania marszruty w przestrzeni stanów, czynności utożsamiane z regułami reakcji *mov1* i *mov2* sprawiają wrażenie niezależnych od siebie, a przez to wykonywanych równolegle (rozpoczętych w tym samym momencie). Jest to oczywiście nieprawdą, ponieważ wykonanie czynności *mov2* jest uzależnione od wcześniejszego przeniesienia obiektu na obszar *AT2*, czyli wykonania czynności *mov1*. Etap 4 algorytmu weryfikacji poprawności modelu MAS sprawdza właśnie dlatego dwa warunki. Pierwszym jest to, czy rozpatrywana czynność może zostać rozpoczęta w aktualnie skonstruowanym stanie systemu. Drugim jest, czy czynność mogłaby zostać wykonana w stanie scenariusza, jaki był na początku rozpatrywanego momentu czasu. W tym przypadku drugi warunek nie został spełniony.

Aby naprawić model należy dokonać dwóch zmian. Pierwszą jest zakwalifikowanie przenoszonego obiektu do grupy obiektów biernych zadania. Tym sposobem wprowadzona zostanie zależność czasowa pomiędzy regułami *mov1* oraz *mov2* i ich wykonanie nie zostanie zaplanowane na ten sam moment czasu. Drugą niezbędną zmianą jest dodanie reguły reakcji reprezentującej czynność, umożliwiającą przeczekanie dronom takiej ilości czasu, jaka jest niezbędna na wykonanie *mov1*. Bez tego dron znajdujący się na obszarze AT3 będzie musiał od razu zakończyć wykonywanie misji, ponieważ nie mogąc uczestniczyć w czynności *mov2* (która nie może zostać rozpoczęta w tak zmodyfikowanym modelu w chwili czasu 0), nie ma nic innego do zrobienia. Wynika to z wymogu ciągłości polityki zachowania dla obiektów zadania, który został zdefiniowany w 3.3.

5.4 Skalowanie projektu

Kolejnym aspektem praktycznego wykorzystania metodyki projektowania roju dronów jest skalowanie już opracowanego z jej użyciem rozwiązania. Mowa tu zarówno o rozszerzaniu modelu „w pionie" jak i „w poziomie". Przez skalowanie w pionie będziemy rozumieć rozbudowę projektu o nowy etap (zadanie realizowane przez projektowany rój). Skalowanie w poziomie będzie rozumiane jako rozszerzanie już istniejącego zadania na większą liczbę elementów (bez względu na to,

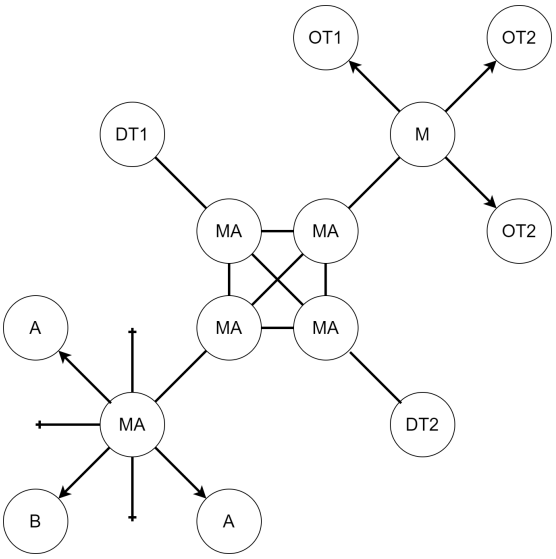
czy są to elementy środowiska czy obiekty).

Opracowana metodyka przedstawiona w rozdziale 3 umożliwia obydwa wyżej wymienione typy modyfikacji projektu. Poniżej zostaną przedstawione propozycje sposobów rozszerzenia wcześniej przygotowanego projektu w obydwu kierunkach.

5.4.1 Skalowanie w poziomie

Celem rozszerzenia modelu w poziomie jest opracowanie sposobu zachowania realizującego zadanie, mające taki sam cel jak dotychczas (lub spełniające takie same wymagania funkcjonalne), ale angażujące więcej obiektów. Model zadania opracowany zgodnie z zaproponowaną w rozdziale 3 metodyką składa się z: definicji zadania, dwugrafowego śledzącego systemu reakcyjnego, systemu przejść oraz przestrzeni stanów. Ponieważ jednym z założeń względem modeli opracowywanych omawianą metodyką jest stała liczba obiektów w zadaniu, zmiana ich liczby pociąga za sobą gruntową zmianę wszystkich elementów modelu. Najmniej wymagająca jest zmiana samej definicji zadania, która jest w formie opisowej. Modyfikacja dwugrafowego systemu reakcyjnego pociąga za sobą zmianę obydwu kolejnych struktur, tj. TTS oraz SS. Jednak dzięki temu, że kolejne elementy modelu są wynikiem przekształcenia elementów wcześniej skonstruowanych, rozszerzenie modelu w poziomie w praktyce ogranicza się do zmiany stanu początkowego (będącego dwugrafem).

Poniżej przedstawiony zostanie przykład, w jaki sposób rozszerzenie modelu w poziomie (wykonane poprzez zmianę stanu początkowego) wpływa na jego elementy. Szczególnie istotne będzie tutaj zwiększenie rozmiaru systemu przejść, który można zdefiniować zarówno jako liczbę dwugrafów jak i samych przejść. Na potrzeby tego przykładu rozpatrywane będzie zadanie przemieszczenia obiektów w magazynie przez parę identycznych dronów. Magazyn składa się z trzech typów obszarów. Pierwszym są obszary magazynowe, pomiędzy którymi drony mogą się swobodnie przemieszczać. W magazynie jest przechowalnia, w której znajdują się obiekty do przeniesienia. Ponadto, są też dwa typy obszarów, do których przenoszone są wspomniane obiekty. Poza samym przeniesieniem obiektów drony wykonujące zadanie mają wrócić do obszaru, w którym znajdowały



Rysunek 5.11: Pierwsza postać stanu początkowego zadania na potrzeby przykładu 5.4.1 - skalowania modelu w poziomie. Nie zawarto na nim wartości identyfikatorów wierzchołków ponieważ nie będą one istotne w tym przykładzie. Powyższy dwugraf należy przez to traktować jako abstrakcyjny.

Tabela 5.12: Interpretacja sterowników użytych w dwugrafach z rysunków 5.11 i 5.12.

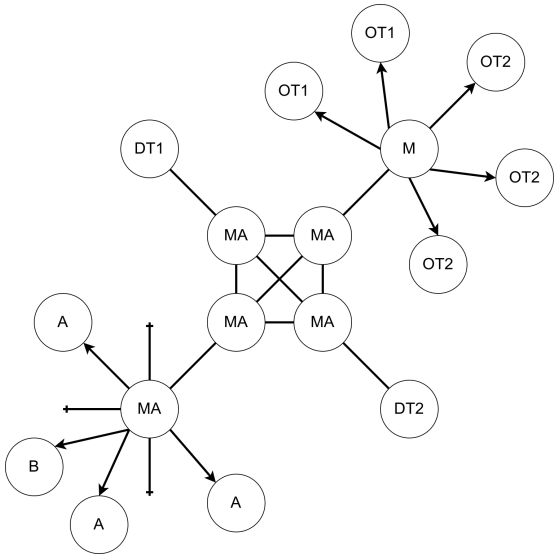
Sterownik	**Obiekt świata rzeczywistego**
A	Dron
MA	Obszar magazynowy - drony mogą się między nimi przemieszczać.
B	Nadajnik - wskazuje obszar magazynowy, do którego drony mają wrócić po przeniesieniu wszystkich obiektów.
M	Przechowalnia - znajdują się w niej obiekty do przeniesienia.
OT1	Obiekt do przeniesienia typu 1.
OT2	Obiekt do przeniesienia typu 2.
DT1	Obszar do składaowania obiektów typu 1.
DT2	Obszar do składaowania obiektów typu 2.

się na początku. W tym celu obszar ten został oznaczony nadajnikiem. Drony mogą pobierać pojedynczo obiekty z przechowalni, jeżeli znajdują się na obszarze magazynowym bezpośrednio do niej przyległym. Mogą również odkładać pobrane wcześniej obiekty, jeżeli znajdują się na obszarze przyległym do obszaru przeznaczonego do składowania obiektu. Każdy z dronów może przemieszczać się niezależnie od drugiego, zarówno będąc w posiadaniu dowolnego typu obiektu jak i „na pusto". Pierwszy rozpatrywany stan początkowy systemu przedstawiono na rysunku 5.11, interpretacja sterowników użytych w tym dwugrafie (abstrakcyjnym) została zamieszczona w tabeli 5.12.

Reguły reakcji dla tego przykładu będą przedstawione w postaci opisowej.

Tabela 5.13: Poglądowa definicja reguł reakcji, zdefiniowanych na podstawie opisu zadania na potrzeby weryfikacji wymagań pozafunkcjonalnych. Wartości w trzeciej kolumnie oznaczają czas potrzebny na wykonanie czynności utożsamianej z regułą reakcji.

Etykieta	**Opis reguły reakcji**	Δt
mov	Przejście drona pomiędzy obszarami magazynu.	1
stay	Dron nie wykonuje żadnej czynności (wykonuje zawis).	1
get1	Dron pobiera obiekt typu 1 z przechowalni.	2
get2	Dron pobiera obiekt typu 2 z przechowalni.	2
set1	Dron odkłada obiekt typu 1 do przeznaczonego dla niego miejsca.	2
set2	Dron odkłada obiekt typu 2 do przeznaczonego dla niego miejsca.	2



Rysunek 5.12: Druga postać stanu początkowego zadania, na potrzeby przykładu skalowania modelu w poziomie.

Jest to podyktowane tym, że ze względu na rozmiary prezentowanego przykładu szczegółowa analiza tego, jak mogą być aplikowane poszczególne reguły do dwugrafów z TBRS/TTS będzie niemożliwa. W tabeli 5.13 zamieszczono opisy wszystkich reguł reakcji dostępnych w opisywanym tutaj systemie wraz z czasem trwania czynności, które reprezentują. Reguła reakcji *stay* odpowiada regule *mov*, dla której obszar początkowy jak i docelowy są tymi samymi.

Przestrzeń stanów zadania omawianego dotychczas ze stanem początkowym jak na rysunku 5.11, skonstruowana przy użyciu programu biblioteki programowej [22] składa się z 666 nieizomorficznych stanów (dwugrafów w TBRS i TTS) oraz 5325 funkcji postępu misji (przejść w TTS).

Rozpatrzmy teraz analogiczne zadanie, składające się z takiego samego zbioru reguł reakcji, ale ze stanem początkowym rozszerzonym tylko o jednego drona, liczbę przenoszonych obiektów typu 1 do dwóch, a liczbę obiektów typu 2 do trzech. Reprezentacja graficzna tego stanu początkowego została przedstawiona na rysunku 5.12. Przestrzeń stanów dla takiego zadania składa się z 5765 stanów, z kolei liczba funkcji postępu misji jest równa 70701. Wyniki te również zostały uzyskane przy użyciu [22].

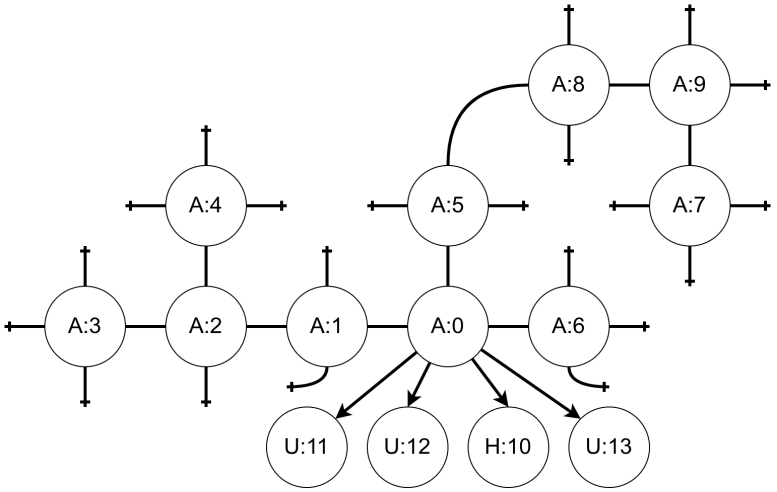
Zarówno liczba przejść, jak i liczba dwugrafów w TTS mogą stanowić o rozmiarze modelu. Są one bezpośrednio powiązane ze złożonością obliczeniową operacji, niezbędnych do wykonania w trakcie wyznaczania polityki zachowania metodą opisaną w 3.3. Złożoność obliczeniowa tej metody została oszacowana w załączniku E.

5.4.2 Skalowanie w pionie

Skalowanie w pionie jest bardziej skomplikowane od rozszerzenia w poziomie, jednak znacznie mniej inwazyjne dla już opracowanych modeli w projekcie. Poniżej zostanie omówiona propozycja, w jaki sposób można rozszerzyć projekt roju, aby realizował kilkuetapowe zadanie. Potrzeba skalowania projektu w pionie może pojawić się zarówno w wyniku dekompozycji zadania postawionego przed rojem na kilka etapów, jak i dodania nowych wymagań funkcjonalnych, w trakcie projektowania już istniejącego systemu. Rozbudowując zadanie do zrealizowania konieczna jest wiedza o tym, co i w jaki sposób jest już realizowane. Można to określić poprzez znalezienie odpowiedzi na następujące pytania:

- Jaki jest stan początkowy nowego zadania?
- Jaki jest stan końcowy poprzedniego zadania ?
- Jaki jest dotychczasowy sposób zachowania (harmonogram działań) obiektów w systemie?

Rozszerzenie zadania do realizacji przez projektowany rój będziemy traktować jako nowe zadanie, które ma zostać wykonane po dotychczas zaplanowanych zadaniach. Skutkuje to koniecznością zdefiniowania wszystkich elementów modelu,



Rysunek 5.13: Stan początkowy systemu do przykładu skalowania projektu w pionie. Wierzchołki typu A reprezentują obszary, typu U drony, a sterownik typu H oznacza bazę, z której drony startują oraz gdzie mają wrócić po wykonaniu pierwszego zadania.

nawet jeżeli większość z nich się powtarza, w którymś z wcześniej wykonywanych zadań. Integracji będą podlegać tylko harmonogramy działań oraz SaT. Każdy harmonogram będzie rezultatem przekształcenia marszruty w przestrzeni stanów, systemu realizującego „zadanie cząstkowe". Poszczególne „modele cząstkowe" mają na celu umożliwić opracowanie sposobu zachowania elementów w systemie, który pozwoli na realizację „zadania cząstkowego". Każdy z tych modeli jest niezależny od pozostałych, z zastrzeżeniem dwóch fragmentów. Pierwszym jest stan początkowy, musi on być taki sam jak stan docelowy (końcowy) modelu realizującego poprzednie zadanie. Z oczywistych powodów ten wymóg nie dotyczy modelu realizującego pierwsze zadanie. Drugim elementem każdego „modelu cząstkowego", który jest zależny od tego realizującego poprzednie zadanie jest początkowa konfiguracja SaT, będąca elementem macierzy $\mathbb{M}_K^0$. Musi być ona identyczna jak ta, która była wynikiem procesu szukania marszruty w przestrzeni stanów, systemu realizującego poprzednie zadanie.

Jako przykład projektowania roju realizującego kilka zadań, omówiony będzie problem pokrycia obszaru przez rój BPL. Różne warianty tego zadania są rozpatrywane w literaturze. Jako uzasadnienie potrzeby wykonania tego typu zadania przyjmiemy, że na mapie znajdują się mobilne punkty pomiarowe, z których dane mają być pobierane przy użyciu dronów. Przyjęto również założenie, że z punktu widzenia odbiorcy końcowego istotne jest, aby pobrane dane przedstawiały stan

wszystkich mobilnych punktów w tym samym (dowolnym) momencie czasu. Z tego powodu każdy z dronów będzie musiał przemieścić się na określoną wcześniej pozycję, poczekać na rozmieszczenie pozostałych dronów i pobrać dane z obszarów, z których może to zrobić bez dalszego przemieszczania się. Po wykonaniu tego zadania drony mają wrócić do bazy. Przyjęto, że pozycja mobilnych punktów pomiarowych jest nieznana w momencie rozpoczynania misji. Ponadto przyjęto założenie, że drony mogą pobierać dane z obszaru, na którym się znajdują oraz wszystkich obszarów bezpośrednio do niego przyległych. Pozostałymi możliwościami działania dronów jest przemieszczanie się pomiędzy obszarami oraz pozostawanie w miejscu, w którym się znajdują. Ta ostatnia czynność ma na celu umożliwienie dronom, które wcześniej dotrą do celu synchronizację z pozostałymi, które potrzebują przebyć dalszą drogę. Przyjęto, że dostępne są trzy drony mogące pobrać dane z nieograniczonej liczby punktów jednocześnie, a stan początkowy systemu w postaci dwugrafu zaprezentowano na rysunku 5.13. Wierzchołki typu A reprezentują obszary, typu U drony, a sterownik typu H oznacza bazę, z której drony startują oraz gdzie mają wrócić po wykonaniu pierwszego zadania. Dla tego przykładu pominięte zostaną definicje funkcji postępu misji i reprezentacje graficzne TBRS, TTS oraz przestrzeni stanów ze względu na ich rozmiary.

Zdefiniowane powyżej zadanie można zdekomponować na dwa etapy. Pierwszym jest rozlokowanie dronów z bazy do miejsc pobrania danych. Każdy dron będzie musiał znajdować się na innym obszarze, aby spełnić wymóg jednoczesnego pobrania danych ze wszystkich punktów pomiarowych. Dokładniej, drony będą musiały znajdować się na obszarach oznaczonych przez wierzchołki o identyfikatorach $\{0, 2, 9\}$, aby były w stanie pobrać dane ze wszystkich obszarów jednocześnie. Reguły reakcji reprezentujące możliwości dronów zostały przedstawione w tabeli 5.14. Reguły *move* i *wait* umożliwiają dronowi odpowiednio przemieszczenie się pomiędzy dowolnymi połączonymi ze sobą obszarami oraz pozostanie na obszarze, na którym aktualnie się znajduje. Przyjmiemy, że czas potrzebny na wykonanie dowolnej z czynności reprezentowanych przez te reguły reakcji zajmuje 1 jednostkę czasu. Reguła *init* inicjuje proces pobierania danych. W trakcie tej czynności obszar, na którym przebywa dron jest oznaczany

wraz z tym dronem. Obszary są oznaczane wierzchołkiem o sterowniku M, a drony sterownikiem I. Będąc zainicjowanym dron nie może się poruszać. Jedyne dwa możliwe działania do wykonania przez drona będącego w takim stanie to oznaczenie kolejnych obszarów (tym razem przyległych do tego, na którym się znajduje) lub pobranie z nich danych. Co bardzo istotne przyjęto, że ten proces jest wykonywany błyskawicznie, przez co czas wykonania tej czynności określony w ramach funkcji postępu misji wynosi 0 jednostek czasu. Reguła reakcji *mark* pozwala na oznaczenie dowolnego przyległego obszaru jako takiego, z którego pobierane będą dane, o ile znajduje się na nim mobilny punkt pomiarowy. Tak samo, jak w przypadku reguły *init*, wykonanie czynności reprezentowanej przez regułę *mark* będzie trwało 0 jednostek czasu. Ostatnia reguła reakcji *download* rozpoczyna proces pobierania danych z obszarów, które zostały oznaczone. Wymagane jest, aby dron pobierający dane był zainicjowany. Przyjęto, że czynność reprezentowana przez tę regułę reakcji zajmuje 2 jednostki czasu.

Stany docelowe (dwugrafy) dla tego zadania to takie, w których każdy wierzchołek reprezentujący obszar zawiera w sobie co najmniej jeden znacznik (wierzchołek typu M).

Drugim etapem zadania będzie powrót dronów do bazy, po pobraniu danych ze wszystkich mobilnych punktów pomiarowych. W celu realizacji tego zadania należy opracować osobny model MAS, którego stanem początkowym będzie wybrany stan docelowy poprzedniego zadania. Możliwe działania dronów, które dokonały pobrania danych z punktów pomiarowych to przemieszczanie się oraz lądowania w bazie. Przyjęto, że obydwie czynności zajmują 1 jednostkę czasu. Reguły reakcji reprezentujące powyższe możliwości przedstawiono w tabeli 5.15.

Przyjmując, że stan docelowy dla zadania pierwszego będzie utożsamiany z dwugrafem przedstawionym na rysunku 5.14, harmonogram działań w reprezentacji graficznej realizujący to zadanie w sposób, który doprowadzi do tego stanu, będzie taki jak na rysunku 5.15. Należy tutaj zwrócić uwagę na część działań, których nie widać na graficznej reprezentacji harmonogramu, z powodu zerowego czasu potrzebnego na ich wykonanie. Dotyczy to czynności reprezentowanych przez reguły reakcji *init* oraz *mark*. W reprezentacji formalnej funkcje postępu misji na podstawie tych reguł reakcji będą oczywiście uwzględnione.

Tabela 5.14: Reguły reakcji dla pierwszego zadania z przykładu 5.4.2. Wszystkie funkcje τ są tożsamościami i mapują wierzchołki tych samych typów w redeksie do wierzchołków w reaktum.

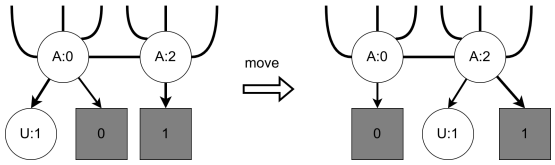
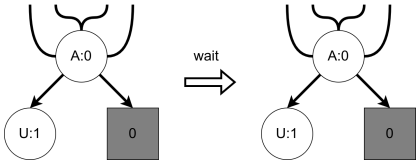
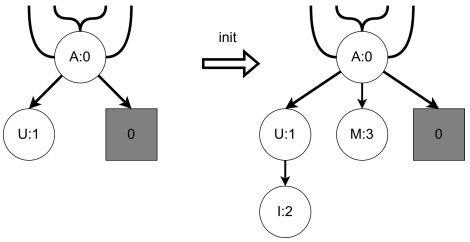
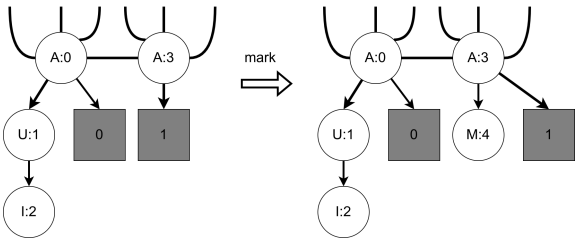
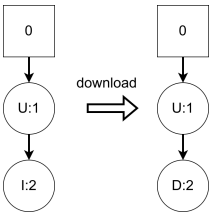
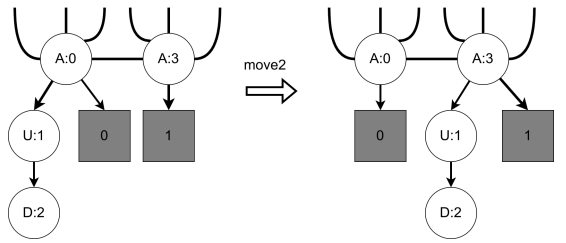
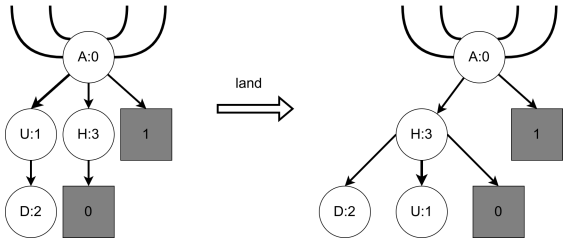
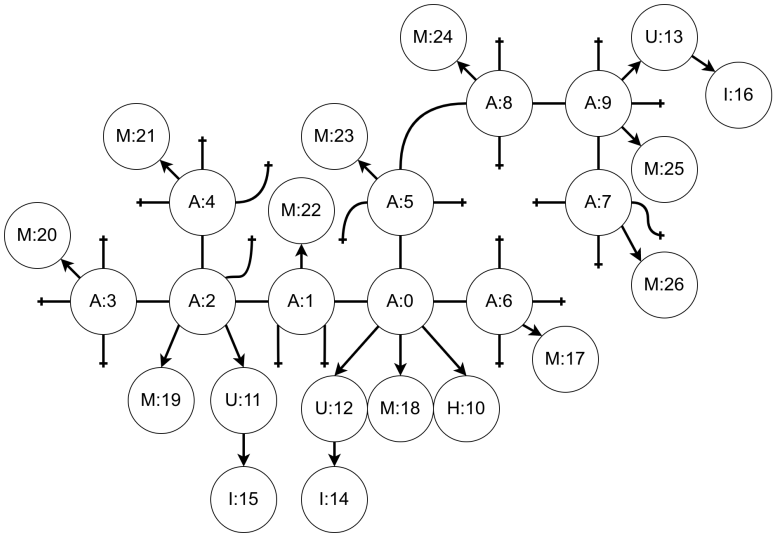
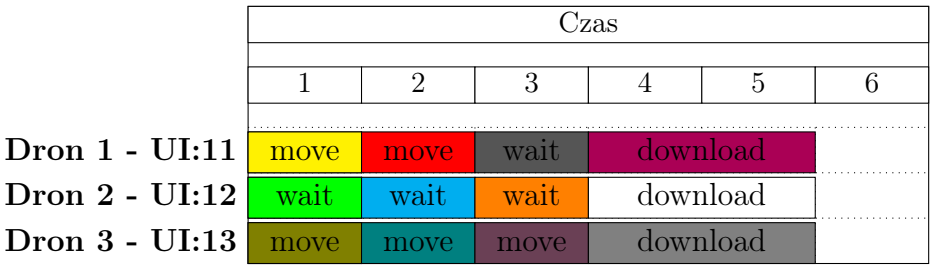
Reguła reakcji	Reprezentacja graficzna
move	
wait	
init	
mark	
download	

Tabela 5.15: Reguły reakcji dla drugiego zadania z przykładu 5.4.2. Wszystkie funkcje śledzenia τ są tożsamościami i mapują wierzchołki tych samych typów w redeksie do wierzchołków w reaktum.

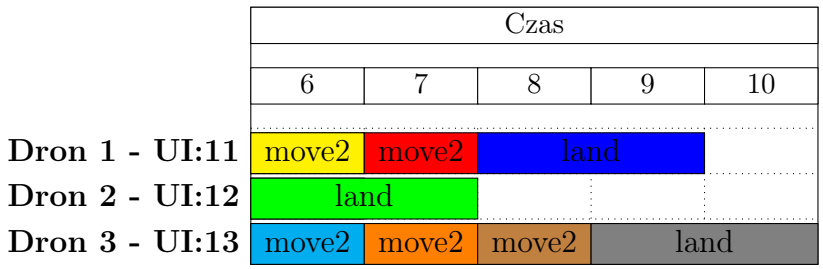
Nazwa reguły reakcji	Reprezentacja graficzna
move2	
land	



Rysunek 5.14: Stan docelowy systemu po wykonaniu pierwszego zadania, w przykładzie 5.4.2.



Rysunek 5.15: Harmonogram działań dla dronów realizujących pierwszy etap zadania, z przykładu 5.4.2. Nie uwzględniono identyfikatorów funkcji postępu misji, ponieważ ich definicje nie zostały przedstawione. Pozostawione nazwy czynności wraz z graficzną reprezentacją stanu docelowego na rysunku 5.14 umożliwiają prześledzenie, który dron i gdzie został przemieszczony.

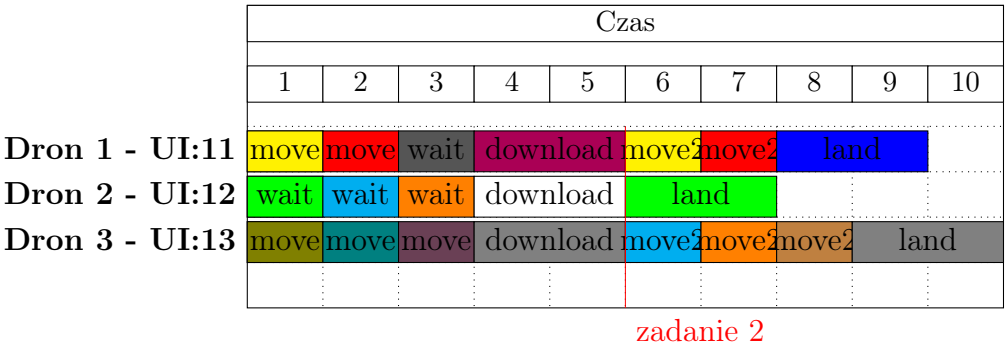


Rysunek 5.16: Harmonogram działań dla dronów realizujących drugie zadanie, z przykładu 5.4.2. Identyfikatory funkcji postępu misji zostały pominięte z tego samego powodu, z którego nie uwzględniono ich na harmonogramie realizującym zadanie pierwsze.

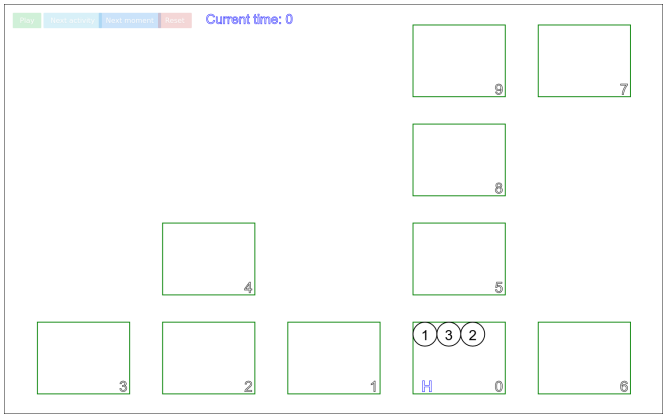
Mając harmonogram działań umożliwiający wykonanie pierwszego zadania, możemy przystąpić do wykonania drugiego zadania, którego celem jest przemieszczenie dronów do bazy. Stan początkowy w modelu systemu dla tego zadania jest taki sam, jak stan docelowy pierwszego zadania tj. taki, jak na rysunku 5.14. Stan docelowy nowego zadania będzie utożsamiany z dwugrafem, w którym wszystkie drony będą znajdowały się wewnątrz wierzchołka reprezentującego bazę. W tym przykładzie czas, w którym wszystkie obiekty zadania, kończą wykonywanie pierwszego zadania jest identyczny. Harmonogram działań, który umożliwi wykonanie tego etapu zadania został przedstawiony na rysunku 5.16.

Dla wieloetapowych zadań wynikiem całego projektu będzie suma kolejnych harmonogramów. W przypadku zadania omawianego w tym rozdziale taki harmonogram został przedstawiony na rysunku 5.17.

Dla systemów, których stany w postaci dwugrafów nie są łatwe w interpretacji, można wykorzystać symulator do zobrazowania przebiegu scenariusza. Przykła-



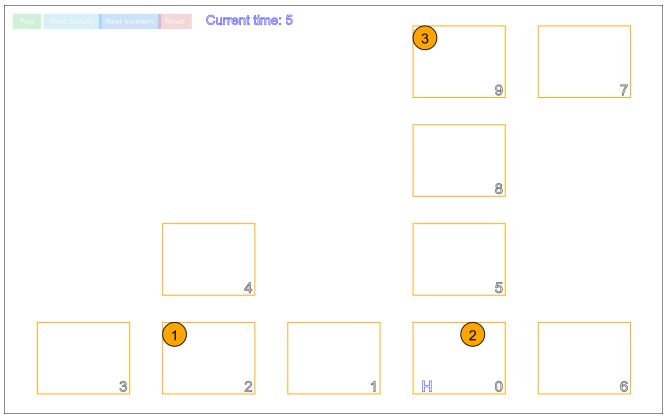
Rysunek 5.17: Harmonogram działań dla dronów realizujących obydwa zadania, z przykładu mającego na celu zademonstrowanie skalowania projektu w pionie.



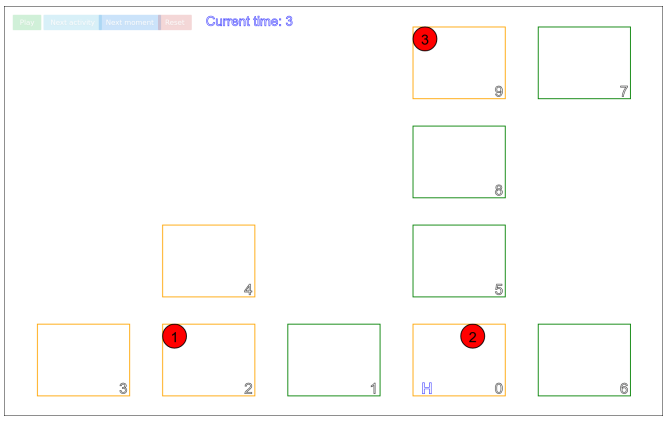
Rysunek 5.18: Stan początkowy zadania dla przykładu skalowania projektu w pionie, zobrazowany przy użyciu narzędzia [27].

dowo, wykorzystując przygotowane do tego narzędzie [27] można w następujący sposób przedstawić przebieg opisywanego w tym rozdziale zadania[1]. Stan początkowy zobrazowany przy użyciu tego narzędzia, odpowiadający dwugrafowi z rysunku 5.13 przedstawiono na rysunku 5.18. Interpretacja tak przedstawionego stanu systemu jest następująca. Zielone prostokąty oznaczają obszary, pomiędzy którymi drony (reprezentowane przez okręgi) mogą się przemieszczać. Liczby wewnątrz obszarów oznaczają ich identyfikatory i są zgodne z identyfikatorami wierzchołków dwugrafu z rysunku 5.13. Obszar, w którym znajduje się niebieska litera *H* oznacza, że znajduje się na nim baza. Każdy obszar może być połączony z czterema innymi, co zostało przedstawione jako przyległość prostokątów. Stan scenariusza w momencie zakończenia pierwszego zadania, w reprezentacji graficznej opracowanej z użyciem symulatora [27] przedstawiono na rysunku 5.19.

[1]Plik *phd_ example.js* z danymi niezbędnymi do przeprowadzenia tej konkretnej symulacji znajduje się w folderze *data*.



Rysunek 5.19: Stan docelowy systemu po wykonaniu pierwszego zadania, w przykładzie skalowania projektowanego zachowania w pionie, zobrazowany przy użyciu symulatora [27].



Rysunek 5.20: Jeden z stanów scenariusza po chwili 3, ale przed chwilą 4, zobrazowany przy użyciu symulatora [27].

Przewagą przedstawienia przebiegu zadania z użyciem opracowanego do tego celu symulatora nad graficzną postacią harmonogramu działań jest to, że można przedstawić czynności o zerowym czasie trwania. Czynności *init* oraz *mark* będą przeprowadzone przed wykonaniem czynności download. Przez to, że zajmują one 0 jednostek czasu, są niewidoczne na harmonogramie z rysunku 5.16. Opracowany symulator pozwala przedstawić wspomniane czynności poprzez wymuszenie postępu symulacji o pojedynczą zaplanowaną czynność. Dla przykładu, jeden ze stanów scenariusza po chwili (momencie czasu) 3, ale przed chwilą 4 został przedstawiony na rysunku 5.20. Przedstawia on sytuację, gdy wszystkie drony zostały zainicjowane, ale tylko dron 1 zdążył oznaczyć przyległe obszary o identyfikatorach 3 i 4.

5.5 Weryfikacja wymagań pozafunkcjonalnych

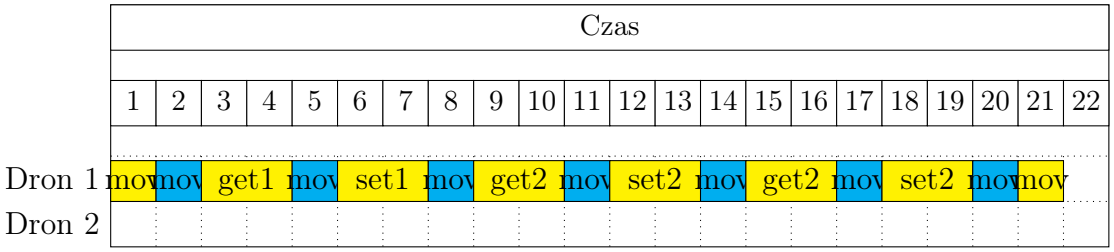
Rozdział ten poświęcony będzie wymaganiom pozafunkcjonalnym stawianym rojom dronów, których zachowanie zostało opracowane przy użyciu metodyki opisanej w rozdziale 3. Poruszone będą tutaj wymagania, których weryfikacja może zostać wykonana przy użyciu jednego z następujących elementów:

- Relacji binarnej *Assignment* oraz funkcji *duration*, będących elementami składowymi każdego harmonogramu.
- Stanów scenariusza, które są wynikiem weryfikacji polityki zachowania, z której konstruowany jest harmonogram działań.
- Funkcji *role*, będącej składnikiem każdego harmonogramu.

Każdy z powyższych elementów umożliwia weryfikację innego typu wymagań pozafunkcjonalnych. W dalszej części tego rozdziału rozpatrzone będą trzy przykłady wymagań, każde możliwe do zweryfikowania przy użyciu jednego z wyżej wymienionych elementów. Zadanie wykorzystane tutaj jest tym samym, które zostało opisane w rozdziale 5.4.1. Rozpatrywany będzie tutaj „mniejszy" wariant z przykładu 5.4.1 tj. taki, w którym przenoszone są w sumie trzy obiekty przez dwa drony. Stan początkowy tego zadania we wspomnianym wariancie został przedstawiony na rysunku 5.11.

Zaczynając od pary składającej się z relacji *Assignment* i funkcji *duration*, umożliwia ona weryfikację wymagań dotyczących czasu trwania misji. Przykładowo, może być to wymóg wykonania zadania w czasie nie większym niż 20 jednostek czasu. Przyjęto, że czas przemieszczenia się dronów pomiędzy obszarami magazynowymi wynosi 1 jednostkę czasu, z kolei na pobranie lub odłożenie wcześniej pobranego obiektu potrzeba 2 jednostki czasu. Założenia te zostały zdefiniowane w tabeli 5.13 definiującej reguły reakcji.

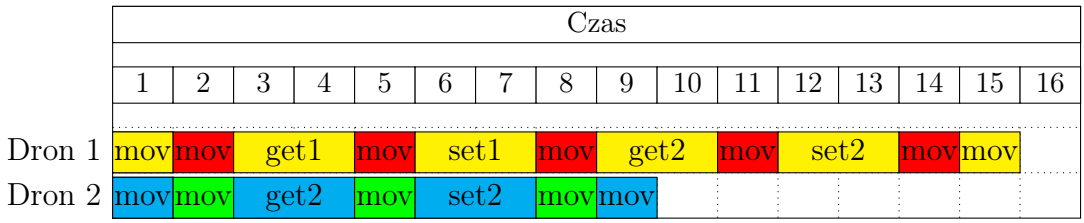
Aby móc zweryfikować, czy to wymaganie zostało spełnione, należy wyznaczyć harmonogram działań dla obydwu dronów. Pierwsze marszruty w przestrzeni stanów, które umożliwiają konstrukcję harmonogramu działań mają długość 15 kroków i żadna z nich nie umożliwia wykonania zadania w czasie poniżej 20 jednostek czasu. Jest tak dlatego, że dowolna kombinacja 15 elementowa z powtórzeniami



Rysunek 5.21: Harmonogram działań dla zadania demonstrującego sposób weryfikacji wymagań pozafunkcjonalnych. Został skonstruowany na podstawie marszruty o długości 15 kroków. Uwzględniono tylko elementy zadania reprezentowane przez wierzchołki typu A.

zbioru 6 elementowego czynności reprezentowanych przez reguły reakcji z tabeli 5.13 realizująca zadanie, nie wykorzysta do tego więcej niż jednego drona. Przykładowy harmonogram opracowany na podstawie jednej z takich marszrut został przedstawiony na rysunku 5.21. W ogólności, aby skrócić czas wykonywania misji można szukać innej kombinacji czynności dla agentów w systemie albo zaangażować większą ich liczbę. Jak wcześniej wspomniano, nie istnieje żadna marszruta o długości 15 kroków, która umożliwia szybsze wykonanie zadania. Oznacza to, że pierwszy sposób poprawy rozwiązania jest tutaj niedostępny. Aby bardziej zaangażować dostępnych agentów w wykonywaną misję należy wydłużyć marszrutę, a przez to dodać nowe czynności do harmonogramu działań. W przypadku omawianego tutaj zadania marszruta umożliwiająca wykonanie misji w czasie krótszym niż 20 jednostek czasu będzie miała długość 18 kroków. Pozwoli ona „przypisać" część czynności mających na celu przeniesienie jednego z obiektów przez pierwszego drona do harmonogramu drugiego, a 3 dodatkowe czynności umożliwią przemieszczenie nowo zaangażowanego drona do przenoszonego obiektu i jego powrót do obszaru oznaczonego nadajnikiem. Przykład harmonogramu działań na podstawie marszruty o długości 18 został przedstawiony na rysunku 5.22.

Wymaganiem pozafunkcjonalnym dla zadania omawianego w tym rozdziale, które można zweryfikować przy użyciu stanów scenariusza jest zagwarantowanie bezkolizyjnego przemieszczania się dronów przenoszących obiekty dowolnego typu. Aby zweryfikować, czy tego typu wymaganie zostało spełnione, można wykorzystać wzorce dwugrafowe, których wystąpienie będzie sprawdzane w każdym dwugrafie reprezentującym stan scenariusza. Gwarancję bezkolizyjnego wykonania zadania otrzymamy, jeżeli w żadnym stanie scenariusza nie zostanie wykryty



Rysunek 5.22: Harmonogram działań dla zadania demonstrującego sposób weryfikacji wymagań pozafunkcjonalnych. Skonstruowany na podstawie marszruty o długości 18 kroków. Uwzględniono tylko elementy zadania reprezentowane przez wierzchołki typu A.

Rysunek 5.23: Wzorce dwugrafowe umożliwiające wykrycie, czy w którymkolwiek stanie scenariusza może dojść do kolizji. Wzorce różnią się typem przenoszonego obiektu.

wzorzec definiujący sytuację kolizji. Wykrycie takiej sytuacji, ze względu na możliwość przenoszenia dwóch typów obiektów wymaga opracowania dwóch wzorców dwugrafowych przedstawionych na rysunku 5.23. W przypadku, gdy dowolny ze wzorców będzie występował w którymkolwiek ze stanów scenariusza, (czyli w stanie scenariusza w dowolnym momencie czasu) to wymagana bezkolizyjność nie będzie zapewniona. Jedynym sposobem poprawy tego typu błędu w harmonogramie działań jest powrót do etapu konstrukcji polityk zachowania i wybór innej, na podstawie której nowy harmonogram zostanie opracowany.

Ostatnim typem wymagań pozafunkcjonalnych omawianych w tym rozdziale będą takie, których spełnialność można sprawdzić wykorzystując funkcję *role*. Przykładem może być wymaganie podobne do pierwszego omawianego w tym rozdziale, ale ograniczające zużycie energii zamiast czasu przez pojedynczego drona. Przyjmijmy, że każdy z dronów jest zasilany akumulatorem o pojemności 1.2Wh, a wykonując czynności z tabeli 5.13 drony można traktować jako urządzenia o mocy:

- *mov* - 500mW

- *get1* oraz *get2* - 750mW
- *set1* oraz *set2* - 1000mW

Ponadto przyjmijmy, że jedna jednostka czasu oznacza 6 minut.

Dla zadania rozszerzonego o powyższe informacje zasadnym jest dodanie wymagania, aby żaden z dronów nie miał zaplanowanych działań, których wykonanie zużywa więcej energii niż pojemność posiadanego akumulatora. Do weryfikacji tak zdefiniowanego wymagania można użyć funkcjonału $f_{\text{power}} : role \to \mathbb{N}$, który przyjmuje jako argument funkcję $role$ i zwraca wartość wyznaczoną przy użyciu formuły:

$$f_{\text{power}} = max_{codom} \circ power$$

gdzie:

- $power : role \to (\mathcal{E} \to \mathbb{N})$ jest operatorem przekształcającym funkcję $role$ w funkcję przypisującą zużycie energii przez elementy zadania. Definicja formalna tego operatora mogłaby mieć postać:

  $$power(role) = \{(e, p) \mid \forall e \in \mathcal{E} \quad p = cons(e, role)\}$$

  Przy czym funkcja $cons : \mathcal{E} \times role \to \mathbb{N}$ wyznacza zużycie energii dla podanego elementu zadania w trakcie misji :

  $$cons(e, role) = \sum_{((e,(f_x,r_x,t)),v) \in role} \begin{cases} 0.5 \cdot 0.1 & \text{jeżeli } r_x == \text{mov} \wedge v == 1 \\ 0.75 \cdot 0.2 & \text{jeżeli } (r_x == (\text{get1} \vee \text{get2})) \wedge v == 1 \\ 1 \cdot 0.2 & \text{jeżeli } (r_x == (\text{set1} \vee \text{set2})) \wedge v == 1 \\ 0 & \text{w pozostałych przypadkach.} \end{cases}$$

  Powyższa definicja opiera się na założeniu, że wierzchołek reprezentujący drona we wszystkich regułach reakcji ma identyfikator o wartości 1 (warunek $v == 1$). Tym sposobem zużycie energii przez drona wykonującego czynność *mov* przez jedną jednostkę czasu będzie równe $0.5 \cdot 0.1 = 0.05$Wh, ale element zadania uczestniczący w tej samej czynności w innej roli ($v \neq 1$) nie będzie zużywał energii;

- $max_{codom} : (\mathcal{E} \to \mathbb{N}) \to \mathbb{N}$ jest funkcjonałem wyznaczającym największą wartość w zbiorze wartości funkcji podanej na wejściu.

Dla powyżej zdefiniowanych elementów oraz dla funkcji *role* harmonogramu przedstawionego w postaci graficznej na rysunku 5.21 funkcjonał f_{power} zwróci wartość 1.5. Wynik ten należy interpretować w taki sposób, że największe zużycie energii przez dowolnego drona w trakcie wykonywania zadania, przy użyciu harmonogramu z rysunku 5.21 będzie wynosiło 1.5Wh, co przekracza pojemność akumulatorów posiadanych przez drony. Tak samo, jak w przypadku wymagań pozafunkcjonalnych ograniczających maksymalny czas wykonywania misji nie istnieje marszruta o długości 15 kroków, na podstawie której harmonogram działań umożliwiłby wykonanie postawionego zadania. Z tego powodu jedynym sposobem poprawienia rozwiązania jest znalezienie innej (w tym przypadku dłuższej) polityki zachowania. Dla funkcji *role* harmonogramu przedstawionego w postaci graficznej na rysunku 5.22 wartość funkcjonału f_{power} jest równa 1.15, co należy interpretować jako maksymalne zużycie energii na poziomie 1.15Wh w trakcie realizacji zadania. Tyle energii zużyje dron 1 według oznaczeń z powyższego rysunku. Dla drona 2 zużycie energii będzie równe 0.7Wh.

5.6 Praktyczne ograniczenia w wykorzystaniu opracowanej metodyki

W tym rozdziale zostały przedstawione oraz omówione niektóre problemy związane z praktycznym wykorzystaniem opracowanej metodyki projektowania, przy użyciu aktualnie dostępnych narzędzi. Zakres trudności, które mogą zostać napotkane został ograniczony do procesu konstrukcji śledzącego systemu przejść. W ramach tego rozdziału dokonano również porównania wydajności opracowanej biblioteki *tracking_bigraph* [22] z aktualnie najbardziej zaawansowanym narzędziem *BigraphER*[2] [76].

W celu analizy praktycznych problemów związanych z wykorzystaniem opracowanej metodyki rozpatrzone zostały cztery warianty pierwszego etapu zadania

[2]Narzędzie to jest publicznie dostępne pod adresem [75].

opisanego w 5.4.2. Wszystkie cztery warianty różniły się tylko stanem początkowym. Liczba oraz same reguły reakcji są jednakowe dla wszystkich wariantów. Należy tu zaznaczyć, że na potrzeby zagadnień omawianych w tym rozdziale reguły reakcji zostały zmienione względem tych przedstawionych w tabeli 5.14. Głównym powodem jest to, że reguła *mark* może zostać zastosowana dowolną liczbę razy do stanu, w którym znajduje się zainicjowany dron. W związku z tym zmodyfikowano sposób reprezentacji obszarów oznaczonych do pobrania danych.

W ramach tego rozdziału wszystkie obszary na początku wykonywania zadania są oznaczone wierzchołkiem typu *M*. Tym sposobem czynności reprezentowane przez reguły reakcji *init* oraz *mark* polegają na „zabraniu" markera (wierzchołka typu *M*), na którym dokonywana jest odpowiednio inicjalizacja drona, lub który jest oznaczany do pobrania danych. W praktyce oznacza to m.in., że redeks reguły *mark* zostaje zamieniony z reaktum. Stan początkowy pierwszego z rozpatrywanych wariantów wspomnianego zadania został zmniejszony względem tego z rysunku 5.13. Usunięto z niego trzy obszary, o identyfikatorach 7,8 i 9, a także jednego drona (wierzchołek o identyfikatorze 13). Dla takiego wariantu zadania rozpatrywanego w rozdziale 5.4.2 liczba stanów w śledzącym systemie przejść, skonstruowanym przy użyciu narzędzia [22] jest równa 541. Korzystając z tego samego narzędzia wyznaczono 1488 przejść pomiędzy 541 stanami. Drugi z rozpatrywanych wariantów jest identyczny z tym omówionym w rozdziale 5.4.2, tj. stan początkowy konstruowanego TTS jest analogiczny[3] do tego z rysunku 5.13. Dla takiego zadania TTS skonstruowany przy użyciu opracowanego na potrzeby tej rozprawy narzędzia składa się z 17774 stanów, pomiędzy którymi zidentyfikowano 72371 przejść. Trzeci z rozpatrywanych wariantów zadania jest rozszerzeniem tego omówionego w rozdziale 5.4.2 o jeden obszar. Nowy obszar w tym wariancie łączy istniejące obszary o identyfikatorach 1,4 i 5. Wykonalność tak zmodyfikowanego zadania nie jest brana pod uwagę w tym podrozdziale. Dla tego wariantu zadania liczba stanów jest równa 53856, a liczba zidentyfikowanych przy użyciu biblioteki [22] przejść między nimi wynosi 235813. Ostatni wariant różni się od trzeciego tym, że rozpatrywany rój jest czteroelementowy. Oznacza to, że między reprezentacją dwugrafową stanu początkowego trzeciego i czwar-

[3]Analogiczny, ponieważ jak zaznaczono na początku rozdziału wszystkim obszarom dodano jako następnik wierzchołek typu *M*.

Tabela 5.16: Warianty zadania z rozdziału 5.4.2 użyte do porównania szybkości bibliotek *bigraph* i *tracking_bigraph* wraz z wynikami pomiarów czasu, potrzebnego na konstrukcję systemów przejść.

wariant zadania	liczba obszarów	liczba dronów	rozmiar sytemu przejść		czas konstrukcji $[s]$		
			l. stanów	l. przejść	LTS	TTS	TTS-P
1	7	2	541	1488	11	2	2
2	10	3	17774	72371	17220	240	62
3	11	3	53856	235813	N/D	1412	354
4	11	4	464947	2855022	N/D	107599	27912

tego wariantu jest różnica tylko jednego wierzchołka. Z kolei śledzący system przejść[4] dla czwartego wariantu omawianego zadania skonstruowany przy użyciu biblioteki [22] składa się z 464947 stanów i 2855022 przejść. Wszystkie powyższe warianty rozpatrywanego zadania zostały podsumowane w tabeli 5.16.

Na podstawie przedstawionych rozmiarów TTS dla różnych wariantów zadania można zauważyć, że rozmiar zadania rośnie znacznie szybciej niż liczba jego elementów. Nie ma również wątpliwości, że wymagane jest komputerowe wspomaganie konstruowania struktur, takich jak śledzące systemy przejść, czy przestrzeń stanów. Zasadnym jest także szukanie sposobu redukcji ich rozmiarów.

Dalsza część tego rozdziału poświęcona jest analizie możliwości wykorzystania w praktyce struktur omówionych w rozdziale 2.5, przy wsparciu opracowanego na potrzeby tej pracy narzędzia [22], a także porównanie[5] wspomnianego narzędzia do wzorcowego [75], które było dla niego podstawą. Dla opisanych wcześniej pierwszych trzech wariantów zadania z rozdziału 5.4.2 czas potrzebny do skonstruowania TTS przy użyciu biblioteki *tracking_bigraph* [22] wynosił odpowiednio 2s, 4min i 23min. Oznacza to, że wyznaczenie TTS systemu składającego się z trzyelementowego roju działającego na 11-elementowym obszarze zajmuje 23 minuty. Opracowane narzędzie umożliwia pominięcie procesu wyznaczania TBRS, jednak podany czas nie uwzględnia znalezienia w skonstruowanym TTS stanu (stanów) docelowego. Nie uwzględnione jest tu również przekształcenie TTS w SS, a co za tym idzie czas potrzebny na znalezienie polityki zachowania realizującej zadanie.

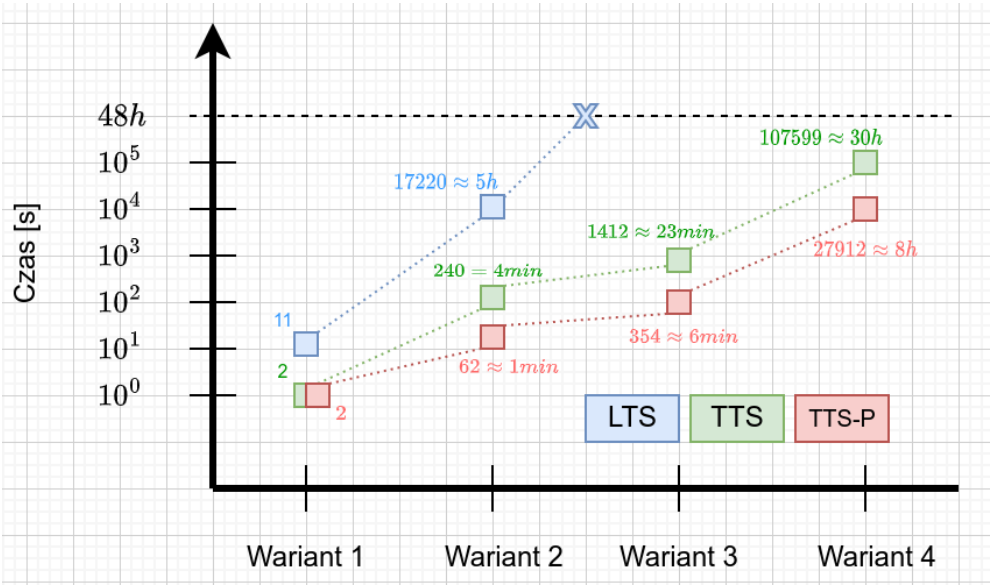
[4]Do zrekonstruowania przedstawionych tutaj wyników potrzebny jest komputer z minimum 13GB pamięci operacyjnej. Ze względu na to oraz na bardzo długi czas potrzebny do przeprowadzenie tego eksperymentu wyniki w postaci plików CSV będących rezultatem działania biblioteki *tracking_bigraph* mogą zostać udostępnione po kontakcie mailowym na adres: *piotr.cybulski@wat.edu.pl*.

[5]Wszystkie omówione w tym rozdziale testy zostały zautomatyzowane przy użyciu skryptu, dostępnego pod adresem *https://github.com/zajer/trs-benchmark* .

W świetle tych wartości zasadnym staje się porównanie opracowanego narzędzia do biblioteki *bigraph*, wchodzącej w skład narzędzia BigraphER, na której było wzorowane. Przed przedstawieniem wyników należy wskazać najistotniejsze różnice między obydwiema bibliotekami programowymi. Kluczową różnicą i powodem, dla którego opracowano *tracking_ bigraph* jest fakt, że biblioteka *bigraph* nie oferuje możliwości wyznaczenia śledzących systemów przejść, a jedynie etykietowane systemy przejść (LTS). Różnica sprowadza się do braku funkcji *par* oraz *res* powiązanych z każdym przejściem. Druga różnica polega na ograniczeniu systemów, dla których może być konstruowany TTS do takich, w których nie występują dwugrafy ze współdzieleniem obszaru nadrzędnego (ang. *bigraphs with sharing*) i których stan początkowy w postaci dwugrafu można zapisać jako $B : \langle 0, \emptyset \rangle \to \langle 0, \emptyset \rangle$. Drugie w wymienionych ograniczeń oznacza, że stany systemu zapisane w postaci dwugrafu nie mogą mieć źródeł, parceli ani zewnętrznych i wewnętrznych połączeń.

W ramach biblioteki *tracking_ bigraph* dokonano dwóch istotnych zmian w implementacji algorytmu konstrukcji systemu przejść względem tej z biblioteki *bigraph*. Pozwoliły one znacząco skrócić czas potrzebny do skonstruowania TTS, względem czasu potrzebnego do skonstruowania LTS, dla identycznych stanów początkowych i analogicznych[6] reguł reakcji. Warto tutaj podkreślić, że biblioteka *tracking_ bigraph* zwraca „obszerniejszy" wynik (bo zawierający funkcje *par* i *res*), a mimo to, potrzebuje znacznie mniej czasu na jego obliczenie. Różnice w czasie potrzebnym na wyznaczenie TTS i odpowiadającemu mu LTS został przedstawiony na rysunku 5.24. Na podstawie uzyskanych wyników można zauważyć, że różnica w czasie potrzebnym na skonstruowanie TTS, a czasem potrzebnym na opracowanie analogicznego LTS rośnie wraz ze wzrostem liczby elementów zadania. Ponadto już dla niewielkich systemów wykorzystanie biblioteki *bigraph* staje się niepraktyczne, ze względu na bardzo długi czas potrzebny na dokonanie niezbędnych obliczeń. Tak znaczące różnice w czasie potrzebnym na opracowanie TTS, a czasem potrzebnym na skonstruowanie analogicznego LTS wynikają z dwóch zmian w implementacji, względem tej w bibliotece *bigraph*. W dalszych rozważaniach należy przyjąć, że szkielet algorytmu konstrukcji (T/L)TS dla oby-

[6]W ramach biblioteki *bigraph* nie są uwzględniane śledzące reguły reakcji.



Rysunek 5.24: Wykres przedstawiający różnicę czasu potrzebnego do konstrukcji analogicznych systemów przejść, przy użyciu biblioteki *bigraph* (LTS) i *tracking_bigraph* (TTS). Wyniki oznaczone kolorem czerwonym przedstawiają czas potrzebny na konstrukcję TTS, wykorzystując 12 rdzeni logicznych (wirtualnych) procesora. Dane charakteryzujące poszczególne warianty przedstawiono w tabeli 5.16.

dwu bibliotek jest identyczny. Przy omawianiu wspomnianego algorytmu należy rozróżnić następujące zbiory dwugrafów, które są rozpatrywane w trakcie konstrukcji systemu przejść:

- zbiór *dwugrafów znanych*, w którym żadna para różnych dwugrafów nie jest równoważna z dokładnością do podstawy (nie są izomorficzne). Zbiór ten można podzielić na dwa podzbiory:
  - *dwugrafy znane niesprawdzone*, czyli takie, których nie zweryfikowano pod kątem wystąpienia redeksów dostępnych reguł reakcji;
  - *dwugrafy znane sprawdzone*, czyli takie, na których ten proces wykonano.
- zbiór *dwugrafów nieznanych*, zawierający dwugrafy, które mogą być nieizomorficzne, do któregoś ze znanych dwugrafów, jak i równoważne któremuś z nich.

Proces konstrukcji systemu przejść dla obydwu bibliotek można opisać następująco:

1. We wszystkich *dwugrafach znanych niesprawdzonych* szukane są wystąpienia redeksów wszystkich reguł reakcji. W ten sposób zbiór ten staje się pusty, a wszystkie elementy do niego należące zostają przeniesione do zbioru *dwugrafów znanych spradzonych.*

2. Każde wystąpienie redeksu jest zastępowane odpowiadającym mu reaktum. Każdy w ten sposób powstały dwugraf jest dodawany do zbioru *dwugrafów nieznanych.*

3. Dla każdego z *dwugrafów nieznanych* sprawdzana jest izomorficzność względem dowolnego z *dwugrafów znanych sprawdzonych.* Ze względu na złożoność tej operacji zostaje ona podzielona na dwa etapy:

   (a) Dla każdego z dwugrafów nieznanych wyliczana jest wartość, która będzie określana mianem "funkcji skrótu z dwugrafu". Są dwa cele tej operacji. Pierwszym jest wyeliminowanie wszystkich *dwugrafów nieznanych*, które na pewno nie są izomorficzne do żadnego *dwugrafu znanego sprawdzonego.* Wszystkie takie dwugrafy są od razu przenoszone do zbioru *dwugrafów znanych niesprawdzonych.* Drugim celem jest ograniczenie liczby dwugrafów ze zbioru znanych sprawdzonych, z którymi sprawdzana jest równoważność.

   (b) Dla wszystkich pozostałych *dwugrafów nieznanych* wykonywane jest sprawdzenie równoważności z każdym dwugrafem ze zbioru znanych sprawdzonych, którego wartość funkcji skrótu jest identyczna z tą dla dwugrafu nieznanego. W wyniku tej operacji każdy z dwugrafów aktualnie nieznanych może zostać zakwalifikowany albo do dwugrafu znanego niesprawdzonego albo do dwugrafu znanego sprawdzonego.

4. Jeżeli zbiór dwugrafów znanych niesprawdzonych jest pusty, to system przejść został skonstruowany. W przeciwnym przypadku następuje powrót do pkt. 1.

Wspomniana wyżej „funkcja skrótu" z dwugrafu dla dowolnych dwóch równoważnych dwugrafów musi zwracać taką samą wartość. Dla dwugrafów nieizomorficznych wartość powinna być różna, ale nie musi.

Kluczowymi operacjami wykonywanymi w trakcie konstrukcji systemów przejść są:

- Znalezienie wszystkich wystąpień redeksów w aktualnie znanych niesprawdzonych dwugrafach.
- Wyznaczenie "funkcji skrótu"dla wszystkich dwugrafów nieznanych.
- Sprawdzenie równoważności dwóch dwugrafów, których wartość funkcji skrótu jest identyczna.

Zmiany dokonane w implementacji procesu konstrukcji systemu przejść, poza rozbudową o wyznaczanie funkcji *par* i *res*, dotyczyły dwóch ostatnich elementów. Pierwsza zmiana polegała na opracowaniu sposobu przekształcenia dowolnego dwugrafu w kolorowany graf skierowany, a następnie wyznaczeniu wartości ze zbioru liczb całkowitych, która identyfikuje tak powstały graf. Definicja algorytmu wyznaczania wartości funkcji skrótu dla dwugrafu została tutaj pominięta.

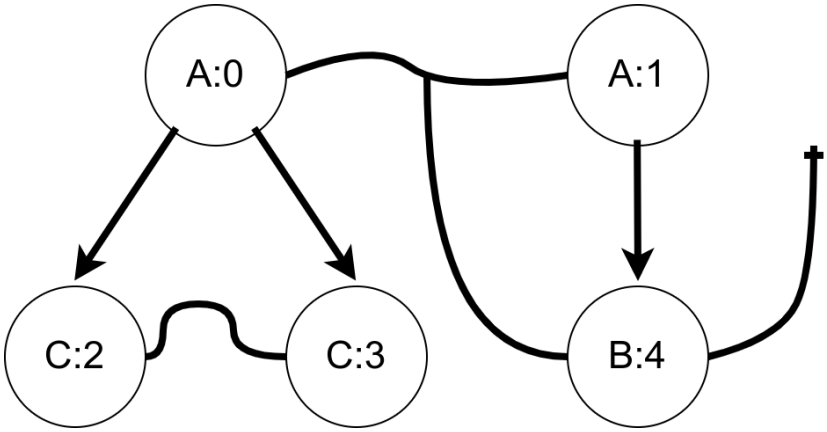
Na podkreślenie zasługuje natomiast zaproponowany sposób, w jaki można przekształcić dwugrafy w kolorowane grafy skierowane. Przekształcenie grafu rozmieszczenia sprowadzono do dodania wierzchołka „źródłowego", dla którego wszystkie wierzchołki nie mające poprzednika są następnikiem. Odwzorowanie hiperkrawędzi z grafu połączeń wykonano zastępując każdą hiperkrawędź wierzchołkiem, którego incydentne wierzchołki grafu połączenia są następnikami. Każdemu wierzchołkowi tak skonstruowanemu grafowi skierowanemu przypisywany jest kolor. Wierzchołki dwugrafu mające ten sam typ sterownika współdzielą kolor na grafie skierowanym. Analogicznie wszystkie wierzchołki reprezentujące hiperkrawędzie mają ten sam kolor. Wierzchołek „źródłowy", który opisano wcześniej również oznaczany jest unikalnym (względem wszystkich dotychczas użytych) kolorem. Przykład przekształcenia dwugrafu w kolorowany graf skierowany przy użyciu opracowanej metody przedstawiono na rysunku 5.25. Druga z istotnych zmian w implementacji mechanizmu konstrukcji systemu przejść polegała na zmianie sposobu weryfikacji, czy dwa dwugrafy są równoważne z dokładnością do podstawy. W bibliotece *bigraph* użyto do tego celu algorytmu opartego na problemie spełnialności (SAT) opisanym w [77]. W przypadku biblioteki *trac-*

king_ bigraph problem ten odpowiada sprawdzeniu izomorficzności dwóch kolorowych grafów skierowanych, a do jego rozwiązania użyto biblioteki *nauty* [60].

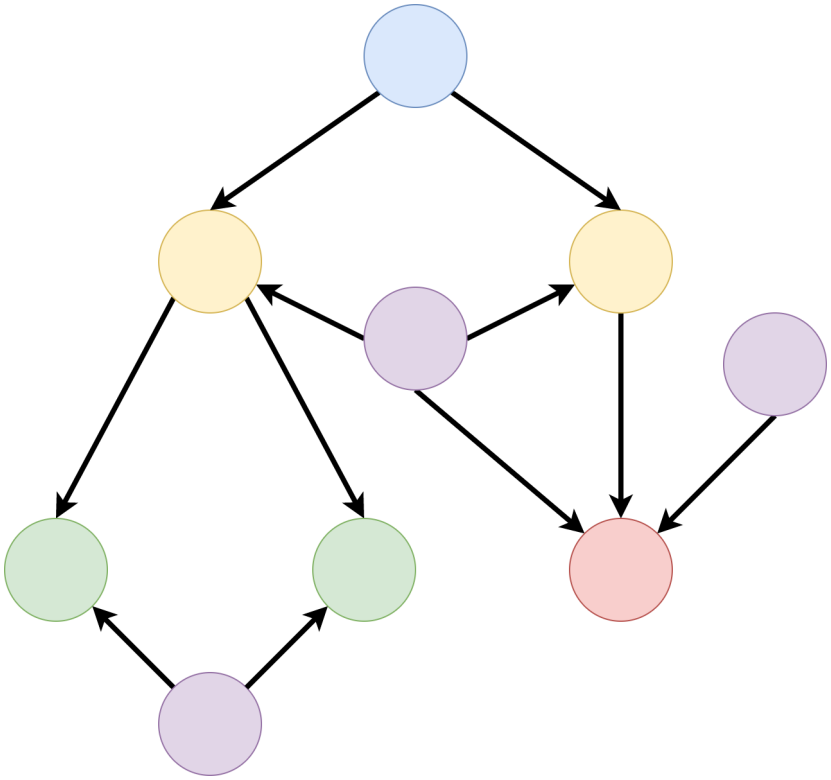
Na uwagę zasługuje jeszcze fakt, że implementacja sposobu konstruowania systemów przejść w bibliotece *tracking_ bigraph* umożliwia do pewnego stopnia zrównoleglenie wykonywanych obliczeń. W aktualnej wersji opracowanej biblioteki zrównolegleniu poddano pierwsze dwa punkty ze schematu zdefiniowanego wcześniej tj. wyszukanie wystąpień redeksów oraz zastąpienie ich przez reaktum. Jest to istotne z tego powodu, że jak widać na rysunku 5.24 czas potrzebny do skonstruowania śledzącego systemu przejść może zostać skrócony nawet o jeden rząd wielkości, względem czasu potrzebnego na skonstruowanie identycznego TTS w sposób jednowątkowy. W przypadku wykonywania obliczeń w sposób jednowątkowy najistotniejszym czynnikiem jest szybkość pojedynczego rdzenia. W sytuacji, gdy wykorzystywanych jest kilka rdzeni procesora ciężar jest rozłożony na trzy elementy: szybkość pojedynczego rdzenia, liczbę dostępnych rdzeni oraz ilość pamięci operacyjnej. W przypadku zrównoleglenia wykonywanych obliczeń w większym stopniu lub zaangażowania większej liczby rdzeni możliwe jest uzyskanie lepszych wyników. Porównując wyniki pomiarów czasu konstrukcji TTS z rysunku 5.24 można zauważyć, że zrównoleglenie obliczeń ma tym większe znaczenie, im większy jest rozmiar rozpatrywanego zadania. Z kolei, porównując wyniki pomiarów czasu potrzebnego do skonstruowania TTS i analogicznego LTS można stwierdzić, że skala zadań rozpatrywanych przy użyciu biblioteki *tracking_ bigraph* jest nieosiągalna dla biblioteki *bigraph*.

W świetle przedstawionych w tym rozdziale rozmiarów projektowanego systemu i czasu potrzebnego na konstrukcję odpowiadającemu mu systemu przejść można zauważyć, że zastosowanie opracowanej metodyki do zadań, w których roje składają się z dużej liczby dronów jest bardzo utrudnione, ze względu na aktualny stan narzędzi wspomagających projektowanie. Ma to szczególnie duże znaczenie przy zadaniach, których sposób rozwiązania wymaga współpracy (jak w przykładzie 5.1) lub gdy liczba elementów roju wpływa na jakość opracowanego rozwiązania (jak w przykładzie 5.5).

Rozwiązaniem tego problemu mogłaby być redukcja liczby przejść w TTS poprzez zdefiniowanie pomocniczych wzorców dwugrafowych. W przypadku zna-



(a) Przykładowy dwugraf przekształcany w ramach biblioteki *tracking_bigraph* w kolorowany graf skierowany.



(b) Kolorowany graf skierowany odpowiadający dwugrafowi z rysunku 5.25a.

Rysunek 5.25: Przykładowy dwugraf wraz z odpowiadającym mu kolorowanym grafem skierowanym, będącym wynikiem przekształceń w ramach biblioteki *tracking_bigraph*.

lezienia, oznaczałyby dwugraf wejściowy jako *znany sprawdzony* bez względu na to, czy można w nim znaleźć wystąpienia redeksów reguł reakcji. Tym sposobem można pośrednio zredukować również liczbę stanów w systemie przejść, ponieważ część z nich może być osiągalna tylko z dwugrafów, które przy użyciu tej metody nie będą miały żadnych przejść. Zaproponowany wyżej sposób nie został zaimplementowany w narzędziu [22]. Biorąc pod uwagę, jak niewielkie zmiany w implementacji względem narzędzia opracowanego przez zespół z Glasgow poprawiły wyniki czasowe można sugerować, że zmiana sposobu wyszukiwania wystąpień redeksów w dwugrafach reprezentujących stany systemu może znacząco przyspieszyć sposób konstrukcji całego TTS.

Rozdział 6

Podsumowanie i kierunki dalszych prac

W pracy przedstawiono metodykę projektowania zachowania dla elementów roju bezzałogowych pojazdów latających oraz metodę weryfikacji modeli wykonanych z jej użyciem. Projektowane zachowanie ma umożliwić realizację zadania stawianego przed rojem.

Podstawą zarówno opracowanej metodyki, jak i metody weryfikacji jest reprezentacja stanów systemu oraz możliwych w nim zmian jako dwugrafów. Umożliwia to przedstawienie zmian, które mogą nastąpić w poszczególnych stanach systemu, przy użyciu dwugrafowego systemu reakcyjnego. Ten z kolei może posłużyć jako podstawa do opracowania śledzącego systemu przejść, który poza powyższym umożliwia wskazanie, które elementy systemu biorą udział w czynnościach prowadzących do pożądanych zmian.

Jedną z kluczowych struktur opracowanych na potrzeby omawianej w tej pracy metodyki jest przestrzeń stanów, którą można skonstruować na podstawie śledzącego systemu przejść. Korzystając ze zdefiniowanej w tej pracy przestrzeni stanów można odwzorować przebieg zadania stawianego przed projektowanym rojem, jako marszrutę pomiędzy dwoma stanami. Taka marszruta, określana polityką zachowania, może zostać przekształcona w harmonogram działań dla wszystkich elementów zadania zaangażowanych w jego wykonanie. Mechanizm umożliwiający opis dynamiki rozpatrywanego systemu wyrażony w języku dwugrafów określany jest jako reguły reakcji. Na potrzeby tej pracy zaproponowano

uproszczoną wersję rozszerzenia tego mechanizmu, zwaną śledzącymi regułami reakcji. Jest to pierwsze znane autorowi tej rozprawy praktyczne wykorzystanie tego rozszerzenia, zarówno w kontekście projektowania zachowania roju dronów, jak i projektowania systemów w ogóle.

Poza metodyką projektowania oraz metodą weryfikacji, przygotowany został zestaw narzędzi [24, 22, 26, 23, 25], którego użyto do weryfikacji wyników. Sposób wykorzystania większości z nich został omówiony w załączniku A. Kluczowym narzędziem jest biblioteka programowa[22] do konstrukcji śledzących systemów przejść. Jest ona rozszerzeniem analogicznej biblioteki służącej do konstrukcji etykietowanych systemów przejść, przygotowanej i rozwijanej przez zespól z Glasgow[75]. W rozdziale 5.6 przedstawiono testy szybkości obydwu bibliotek. Pomimo tego, że śledzące systemy przejść są strukturami „obszerniejszymi" niż etykietowane systemy przejść, to przygotowane na potrzeby tej pracy oprogramowanie jest o kilka rzędów wielkości szybsze niż odpowiednik z Glasgow. Umożliwia ono ponadto wykonywanie części obliczeń związanych z konstrukcją TTS w sposób równoległy, a to jeszcze bardziej skraca czas potrzebny na ich konstrukcję, co zademonstrowano na rysunku 5.24. Skonstruowano również symulator [27], który pozwala zobrazować przebieg misji na podstawie harmonogramu działań. Ze względu na złożoność tego narzędzia nie omówiono sposobu jego wykorzystania.

Przedstawione w rozdziale 5 wyniki skupiały się wokół kluczowych aspektów związanych z projektowaniem zachowania roju BPL. Uwzględniono w nich zadania, które do satysfakcjonującego wykonania wymagają współpracy pomiędzy dronami oraz takie, które mogą być rozwiązane na kilka sposobów. Ponadto określono sposób, w jaki można zweryfikować spełnienie wymagań funkcjonalnych oraz pozafunkcjonalnych. Omówiono na szczegółowych przykładach, w jaki sposób opracowana metoda weryfikacji sprawdza przygotowany model, zarówno w przypadku, gdy model ten jest poprawny, jak i niepoprawny. W świetle przedstawionych wyników z rozdziału 5.6 przedstawiających rozmiary projektowanego systemu w zależności od liczby elementów zadania można zauważyć, że zastosowanie opracowanej metodyki do zadań, w których roje składają się z dużej liczby elementów jest utrudniona, ze względu na obecny stan narzędzi komputerowych.

Za największe osiągnięcie przedstawionej rozprawy uznano ścisłe powiązanie struktur formalnych, używanych na poszczególnych etapach metodyki. Umożliwia to algorytmizację wykonywania kolejnych etapów projektowania zachowania, a to z kolei pozwala w istotnym stopniu zautomatyzować cały proces projektowania zachowania. W rozdziale 3.5 przedstawiającym całościowy obraz opracowanej metodyki przedstawiono wszystkie etapy projektowania wykonywane w ramach pojedynczego etapu zadania. Dokonano tam podziału na etapy projektowania (tam określane fazami), które mogą zostać zautomatyzowane oraz te, które wymagają zaangażowania projektanta. Możliwość automatyzacji procesu projektowania potwierdzono wskazując, które z przygotowanych narzędzi realizuje poszczególne etapy projektowania. Biorąc pod uwagę, że 6 z 9 etapów (faz) metodyki może zostać zalgorytmizowanych można twierdzić, że metodyka jest wysoce zautomatyzowana.

Na podstawie przedstawionych wyników eksperymentów, mających na celu sprawdzenie używalności rezultatów pracy w domenie rojów BPL, można wyciągnąć następujące wnioski:

- Rozmiar systemu mierzony liczbą stanów albo liczbą możliwych w nim zmian rośnie znacznie szybciej, niż liczba elementów rozpatrywanego zadania. Jest to szczególnie widoczne analizując zmianę rozmiarów systemu (mierzoną liczbą stanów i przejść w TTS) wraz ze wzrostem liczby elementów dwugrafu, reprezentującego stan początkowy. Szczególnie zauważalna jest różnica rozmiarów systemu przedstawiona w rozdziale 5.6, pomiędzy zadaniem tam opisanym w wariancie 3 i 4. Dwugrafy reprezentujące stany początkowe tych wariantów różnią się tylko jednym wierzchołkiem, podczas gdy liczba stanów wariantu 4 jest ośmiokrotnie większa niż ta wariantu 3. Z kolei liczba przejść wariantu 4 jest ponad dziesięciokrotnie większa niż liczba przejść wariantu 3.

- Rozpatrywanie problemów związanych z modelowaniem zadań stawianych rojom dronów jest możliwe przy użyciu pojęć oferowanych przez dwugrafy. Można dzięki nim modelować zależności pomiędzy elementami zadania, które mają charakter hierarchiczny (graf rozmieszczenia), jak i takie zależności, które nie wyróżniają żadnego elementu w nie wchodzącego (graf

połączeń). Dynamika tak modelowanych systemów jest przedstawiana przy użyciu reguł reakcji. Pozwala to na definiowanie możliwych zmian w systemie przy użyciu par dwugrafów, reprezentujących sytuacje typu: co musi się stać (redeks), aby coś nastąpiło (reaktum).

- Możliwe jest projektowanie sposobu działania systemów wielorobotowych w taki sposób, aby akceptacja wymagań funkcjonalnych przez interesariuszy wymagała od nich tylko rozumienia pojęć, związanych z dwugrafowymi systemami reakcyjnymi w notacji graficznej.

Do wad uzyskanych rezultatów pracy należy zaliczyć niektóre założenia formułowane względem projektowanych systemów. Przede wszystkim stałą liczbę obiektów w zadaniu (w rozumieniu definicji z rozdziału 3.1), a także pełną kontrolę nad agentami. Za wadę można też uznać to, że opracowana metoda weryfikacji modeli pozwala zdiagnozować brak wystąpienia określonego rodzaju błędów dla pojedynczej polityki zachowania, a nie uniwersalną poprawność modelu. Drugi typ poprawności gwarantowałby, że wszystkie opracowane polityki zachowania będą używalne.

Na podstawie przeprowadzonych eksperymentów i rozważań teoretycznych zidentyfikowano kilka kierunków prac, których wyniki mogłyby znacząco poszerzyć zakres zastosowań opracowanej metodyki, jak i usprawnić możliwości jej wykorzystania w obszarach, które zostały omówione w tej rozprawie. Przede wszystkim jest to rozszerzenie definicji przestrzeni stanów o uwzględnienie agentów, którzy nie są sterowalni. Pozwoli to znacząco powiększyć zakres modelowanych zadań, nie tylko stawianych bezzałogowym pojazdom latającym. Istotnym usprawnieniem rozszerzającym zastosowanie metodyki byłaby także modyfikacja przestrzeni stanów, umożliwiająca przedstawienie zmiennej liczby obiektów w trakcie wykonywania zadania. Należy również podkreślić, że przyspieszenie sposobu wyszukiwania wzorców dwugrafowych (w szczególności redeksów reguł reakcji) znacząco usprawniłoby konstrukcję śledzących dwugrafowych systemów reakcyjnych, czy śledzących systemów przejść.

Problemem natury technicznej, którego rozwiązanie pozwoliłoby przyspieszyć wykonywanie obliczeń związanych z wyznaczaniem polityki zachowania byłby sposób ograniczenia liczby stanów uwzględnianych w TBRS i TTS, przy jedno-

czesnym zachowaniu tych, które umożliwią konstrukcję polityki zachowania. Podobnym problemem, którego rozwiązanie zmniejszyłoby wymagania pamięciowe względem komputerów, na których wykonywane byłyby obliczenia, jest sposób redukcji liczby marszrut w trakcie poszukiwania polityki zachowania do stanu docelowego. Ostatnim kierunkiem prac w tej kategorii byłoby użycie przyspieszenia sprzętowego (w szczególności kart graficznych), przy wykonywaniu obliczeń związanych zarówno z konstrukcją śledzących systemów przejść, jak i przy wyznaczaniu polityki zachowania.

Przechodząc do kierunków dalszych prac związanych ze sposobem weryfikacji modeli, opracowanych przy użyciu metodyki w jej aktualnej postaci, bez wątpienia najwyższy priorytet miałaby formalna metoda oceny sterowników. Jej celem byłoby określenie, które z nich nie mogą być zakwalifikowane do elementów otoczenia zadania. Alternatywą mogłaby być metoda, która wskazywałaby na sterowniki, które muszą być zakwalifikowane jako obiekty zadania. Usprawnieniem byłoby również opracowanie sposobu konstrukcji harmonogramu działań, niezależne od weryfikacji modelu.

Wykaz skrótów i oznaczeń

$[\, L \,]$ symbol Iversona przekształcający zdanie logiczne L w wartość liczbową. 21

$\circ$ złożenie funkcji lub dwugrafów o kompatybilnych interfejsach. 21, 31

$\cong$ przynależność do tego samego zbioru dwugrafów abstrakcyjnych. 31

$\mathcal{B}$ zbiór dwugrafów w TBRS. 43, 181

$\mathcal{R}$ zbiór reguł reakcji w TBRS. 43, 181

$\uplus$ operator łączenia funkcji i rozłącznych zbiorów. 18, 21

codom funkcjonał zwracający przeciwdziedzinę funkcji. 20

dom funkcjonał zwracający dziedzinę funkcji. 20

i_X funkcja tożsamościowa. 21

m, n zbiory porządkowe. 19

max funkcja zwracająca większy z dwóch argumentów. 21

min funkcja zwracająca mniejszy z dwóch argumentów. 21

$\overrightarrow{X}\!\!\downarrow (x)$ funkcja przekształcająca element x zbioru X w liczbę naturalną. 20

τ funkcja śledząca. 41

BPL bezzałogowoy pojazd latający. 5

MAS ang. *multi-agent system* - system wieloagentowy. 3, 49

MRS ang. *multi-robot system* - system wielorobotowy. 3

RS ang. *robotic swarm* - rój robotów. 4

SAR ang. *Search And Rescue* - poszukiwanie i ratownictwo. 8

SaT element zbioru $\mathbb{N}_1^2 \times \cdots \times \mathbb{N}_{n_o}^2$, gdzie n_o to liczba obiektów w zadaniu. 53

SS ang. *state space* - przestrzeń stanów. 52

TBRS ang. *tracking bigraphical reactive system* - śledzący dwugrafowy system reakcyjny. 43

TTS ang. *tracking transition system* - śledzący system przejść. 46

UAV ang. *unmanned aerial vehicle* - bezzałogowy pojazd latający. 5

UI unikalny identyfikator. 61, 73, 75

Słownik terminów

agent Obiekt czynny w zadaniu lub byt mający możliwości obserwacji oraz wchodzenia w interakcję z otoczeniem. 3, 51

dwugraf abstrakcyjny Zbiór dwugrafów konkretnych równoważnych z dokładnością do zwięzłych podstaw. 30

dwugraf konkretny Dla określonej sygnatury, para składająca się z grafu rozmieszczenia oraz grafu połączeń. 27

element zadania Element świata rzeczywistego istotny w rozpatrywanym zadaniu. 50

funkcja śledząca Funkcja przypisująca wierzchołki redeksu wierzchołkom reaktum. Służy do wskazania, które elementy zadania po wykonaniu czynności reprezentowanej przez regułę reakcji odpowiadają elementom zadania sprzed jej wykonania. 41

harmonogram działań Struktura matematyczna pozwalająca określić, które elementy zadania biorą udział w jakich czynnościach, kiedy, a także jaka jest ich rola w każdym działaniu. 61

konfiguracja czasowo-strukturalna Zbiór n-tek uporządkowanych, gdzie n to suma obiektów biernych i czynnych w systemie. 53

misja Sposób realizacji zadania. 51

obiekt bierny Element zadania, który nie inicjuje czynności w świecie rzeczywistym. 51

obiekt czynny Element zadania, który może inicjować czynności w świecie rzeczywistym. 51

otoczenie realizowanego zadania Elementy zadania, które nie mogą inicjować czynności w świecie rzeczywistym, dla których upływ czasu jest nieistotny. 51

podstawa dwugrafu Zbiór zawierający identyfikatory wierzchołków oraz połączeń zamkniętych dwugrafu. 30

polityka zachowania Marszruta w przestrzeni stanów. Polityka zachowania jest podstawą do opracowania harmonogramu działań dla wszystkich elementów zadania. 56

przestrzeń stanów Zbiór unikalnych stanów MAS wraz z informacją, w jaki sposób może dojść do ich zmian. 52

reguła reakcji Możliwa zmiana w systemie modelowanym z użyciem dwugrafów. 40

rój robotów System wielorobotowy, w którym sposób działania opiera się na założeniu, że może zostać opracowany w oparciu o lokalne interakcje. 4

scenariusz Misja z wykorzystaniem określonej polityki zachowania. 57, 92

sygnatura dwugrafu Zbiór wszystkich sterowników wraz z funkcją przypisującą każdemu z nich liczbę portów. 25

system wieloagentowy Zbiór agentów działających we współdzielonym środowisku. 3, 49

system wielorobotowy System wieloagentowy, gdzie agenci są robotami. 3

zadanie Zbiór elementów świata rzeczywistego wraz z ich możliwościami, stanem początkowym oraz końcowym(i). 50

zbiór porządkowy Zbiór, który ma postać $m = \{0, 1, \ldots, m-1\}$. 18

śledzący dwugrafowy system reakcyjny Zbiór śledzących reguł reakcji wraz ze zbiorem uziemionych dwugrafów. 40

śledzący system przejść Struktura umożliwiająca analizę uczestnictwa w czynnościach reprezentowanych przez reguły reakcji. 46

Bibliografia

[1] Harold Abelson i in. „Amorphous Computing". W: *Commun. ACM* 43.5 (maj 2000), s. 74–82. ISSN: 0001-0782. DOI: 10.1145/332833.332842.

[2] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. eng. 1985.

[3] Noa Agmon, Sarit Kraus i Gal A. Kaminka. „Multi-robot perimeter patrol in adversarial settings". W: *2008 IEEE International Conference on Robotics and Automation*. 2008, s. 2339–2345. DOI: 10.1109/ROBOT.2008.4543563.

[4] Blair Archibald, Muffy Calder i Michele Sevegnani. „Conditional Bigraphs". W: *Graph Transformation*. Red. Fabio Gadducci i Timo Kehrer. Cham: Springer International Publishing, 2020, s. 3–19. ISBN: 978-3-030-51372-6. DOI: 10.1007/978-3-030-51372-6_1.

[5] Blair Archibald i in. „BigraphTalk: Verified Design of IoT Applications". W: *IEEE Internet of Things Journal* 7.4 (2020), s. 2955–2967. DOI: 10.1109/JIOT.2020.2964026.

[6] Jonathan Bachrach, James Mclurkin i Anthony Grue. „Protoswarm: A language for programming multi-robot systems using the amorphous medium abstraction". W: t. 2. Sty. 2008, s. 1175–1178. URL: http://www.eecs.berkeley.edu/~jrb/papers/protoswarm-aamas-2008.pdf.

[7] Ilker Bekmezci, Ozgur Sahingoz i ßamil Temel. „Flying ad-hoc networks (FANETs): a survey". W: *Ad Hoc Networks* 11 (maj 2013), s. 1254–1270. DOI: 10.1016/j.adhoc.2012.12.004.

[8] Patrick Benavidez i in. „Design of a home multi-robot system for the elderly and disabled". W: *2015 10th System of Systems Engineering Conference (SoSE)*. 2015, s. 392–397. DOI: 10.1109/SYSOSE.2015.7151907.

[9] Steve Benford i in. „On Lions, Impala, and Bigraphs: Modelling Interactions in Physical/Virtual Spaces". W: *ACM Trans. Comput.-Hum. Interact.* 23.2 (maj 2016), 9:1–9:56. ISSN: 1073-0516. DOI: 10.1145/2882784.

[10] Gerardo Beni. „From Swarm Intelligence to Swarm Robotics". W: *Swarm Robotics*. Red. Erol Şahin i William M. Spears. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 1–9. ISBN: 978-3-540-30552-1. DOI: 10.1007/978-3-540-30552-1_1.

[11] Janusz Będkowski i in. „Badanie systemu robotów mobilnych w aplikacji RISE". W: *Pomiary Automatyka Robotyka* 15.171 (lut. 2011), s. 323–332. URL: http://www.par.pl/en/Yearbooks/2011/2-2011/Badanie-systemu-robotow-mobilnych-w-aplikacji-RISE.

[12] Jan Dyre Bjerknes i Alan F. T. Winfield. „On Fault Tolerance and Scalability of Swarm Robotic Systems". W: *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Red. Alcherio Martinoli i in. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, s. 431–444. DOI: 10.1007/978-3-642-32723-0_31.

[13] D Bozhinoski i M Birattari. „Towards an integrated automatic design process for robot swarms [version 1; peer review: 2 approved]". W: *Open Research Europe* 1.112 (2021). DOI: 10.12688/openreseurope.14025.1.

[14] Manuele Brambilla i in. „Property-Driven Design for Robot Swarms: A Design Method Based on Prescriptive Modeling and Model Checking". W: *ACM Trans. Auton. Adapt. Syst.* 9.4 (grud. 2014). ISSN: 1556-4665. DOI: 10.1145/2700318.

[15] Manuele Brambilla i in. „Swarm robotics: a review from the swarm engineering perspective". W: *Swarm Intelligence* 7 (2012), s. 1–41. DOI: 10.1007/s11721-012-0075-2.

[16] Matthias R. Brust i in. „Swarm-based counter UAV defense system". W: *Discov. Internet Things* 1 (2021). DOI: 10.1007/s43926-021-00002-x.

[17] Francesco Bullo, Jorge Cortés i Sonia Martínez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009. ISBN: 9781400831470. DOI: 10.1515/9781400831470. URL: https://doi.org/10.1515/9781400831470.

[18] Muffy Calder i in. „Real-time verification of wireless home networks using bigraphs with sharing". W: *Science of Computer Programming* 80 (2014), s. 288–310. ISSN: 0167-6423. DOI: 10.1016/j.scico.2013.08.004.

[19] Y. Uny Cao, Andrew B. Kahng i Alex S. Fukunaga. „Cooperative Mobile Robotics: Antecedents and Directions". W: *Robot Colonies*. Red. Ronald C. Arkin i George A. Bekey. Boston, MA: Springer US, 1997, s. 7–27. DOI: 10.1007/978-1-4757-6451-2_1.

[20] IAIN D. COUZIN i in. „Collective Memory and Spatial Sorting in Animal Groups". W: *Journal of Theoretical Biology* 218.1 (2002), s. 1–11. ISSN: 0022-5193. DOI: 10.1006/jtbi.2002.3065.

[21] Valentino Crespi, A. G. Galstyan i Kristina Lerman. „Top-down vs bottom-up methodologies in multi-agent system design". W: *Autonomous Robots* 24 (2008), s. 303–313. DOI: 10.1007/s10514-007-9080-5.

[22] Piotr Cybulski. *Biblioteka programowa do generowania TTS*. URL: https://github.com/zajer/trs.

[23] Piotr Cybulski. *Biblioteka programowa do konstruowania polityk zachowania.* URL: https://github.com/zajer/trs-ssp.

[24] Piotr Cybulski. *Narzędzie do weryfikacji modeli MAS na podstawie polityki zachowania.* https://github.com/zajer/trs-ssp-verif.

[25] Piotr Cybulski. *Narzędzie do zobrazowania przebiegu zadania.* URL: https://github.com/zajer/trs-ssp-frontend.

[26] Piotr Cybulski. *Oprogramowanie do przekształcania TTS w przestrzeń stanów.* URL: https://github.com/zajer/trs-ssp-bridge.

[27] Piotr Cybulski. *Symulator systemów zmieniających się na podstawie harmonogramów działań.* URL: https://github.com/zajer/schvis.

[28] Piotr Cybulski i Zbigniew Zieliński. „Design and Verification of Multi-Agent Systems with the Use of Bigraphs". W: *Applied Sciences* 11.18 (2021). ISSN: 2076-3417. DOI: 10.3390/app11188291. URL: https://www.mdpi.com/2076-3417/11/18/8291.

[29] Piotr Cybulski i Zbigniew Zieliński. „UAV Swarms Behavior Modeling Using Tracking Bigraphical Reactive Systems". W: *Sensors* 21.2 (2021). ISSN: 1424-8220. DOI: 10.3390/s21020622. URL: https://www.mdpi.com/1424-8220/21/2/622.

[30] Ahmed Taki Eddine Dib, Kamel Barkaoui i Zaïdi Sahnoun. „Specification and verification of reconfigurable multi-agent system architectures". W: *Multiagent Grid Syst.* 12 (2016), s. 105–124. DOI: 10.3233/MGS-160246.

[31] Ahmed Taki Eddine DIB i Ramdane MAAMRI. „Bigraphical Modelling and Design of Multi-Agent Systems". W: *2021 7th International Conference on Computing and Artificial Intelligence*. ICCAI 2021. Tianjin, China: Association for Computing Machinery, 2021, s. 365–371. ISBN: 9781450389501. DOI: 10.1145/3467707.3467762. URL: https://doi.org/10.1145/3467707.3467762.

[32] Gregory Dudek i in. „A taxonomy for multi-agent robotics". W: *Auton. Robots* 3 (grud. 1996), s. 375–397. DOI: 10.1007/BF00240651.

[33] Markus Eich i in. „A Robot Application for Marine Vessel Inspection". W: *Journal of Field Robotics* 31.2 (2014), s. 319–341. DOI: 10.1002/rob.21498.

[34] A. Farinelli, L. Iocchi i D. Nardi. „Multirobot systems: a classification focused on coordination". W: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.5 (2004), s. 2015–2028. DOI: 10.1109/TSMCB.2004.832155.

[35] Gianpiero Francesca i M. Birattari. „Automatic Design of Robot Swarms: Achievements and Challenges". W: *Frontiers Robotics AI* 3 (2016), s. 29. DOI: 10.3389/frobt.2016.00029.

[36] Gianpiero Francesca i in. „AutoMoDe: A novel approach to the automatic design of control software for robot swarms". W: *Swarm Intelligence* 8 (2014), s. 89–112. DOI: 10.1007/s11721-014-0092-4.

[37] Amal Gassara, Ismael Bouassida i Mohamed Jmaiel. „A Tool for Modeling SoS Architectures Using Bigraphs". W: *Proceedings of the Symposium on Applied Computing*. SAC '17. Marrakech, Morocco: Association for Computing Machinery, 2017, s. 1787–1792. ISBN: 9781450344869. DOI: 10.1145/3019612.3019802.

[38] Arnab Ghosh i in. *Multi-robot Cooperative Box-pushing problem using Multi-objective Particle Swarm Optimization Technique*. 2012. DOI: 10.48550/ARXIV.1206.5170. URL: https://arxiv.org/abs/1206.5170.

[39] Jakub Gutenbaum. *Modelowanie matematyczne systemów*. Oficyna Wydawnicza EXIT, 2003. ISBN: 83-87674-53-2.

[40] Maki K. Habib. „Service Robots and Humanitarian Demining". W: *Mobile Robots*. Red. Aleksandar Lazinica. Rijeka: IntechOpen, 2006. Rozd. 23. DOI: 10.5772/4704.

[41] Maki K. Habib i Yvan Baudoin. „Robot-Assisted Risky Intervention, Search, Rescue and Environmental Surveillance". W: *International Journal of Advanced Robotic Systems* 7.1 (2010), s. 10. DOI: 10.5772/7249.

[42] Heiko Hamann. *Swarm Robotics: A Formal Approach*. 2018. ISBN: 978-3-319-74526-8. DOI: 10.1007/978-3-319-74528-2.

[43] Ken Hasselmann, Frédéric Robert i Mauro Birattari. „Automatic Design of Communication-Based Behaviors for Robot Swarms". W: *Swarm Intelligence*. Red. Marco Dorigo i in. Cham: Springer International Publishing, 2018, s. 16–29. ISBN: 978-3-030-00533-7. DOI: 10.1007/978-3-030-00533-7_2.

[44] Samira Hayat, Evşen Yanmaz i Raheeb Muzaffar. „Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint". W: *IEEE Communications Surveys & Tutorials* 18.4 (2016), s. 2624–2661. DOI: 10.1109/COMST.2016.2560343.

[45] Daojing He i in. „An Effective Countermeasure Against UAV Swarm Attack". W: *IEEE Network* 35.1 (2021), s. 380–385. DOI: 10.1109/MNET.011.2000380.

[46] Luca Iocchi, Daniele Nardi i Massimiliano Salerno. „Reactivity and Deliberation: A Survey on Multi-Robot Systems". W: *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, s. 9–32. ISBN: 978-3-540-44568-5. DOI: 10.1007/3-540-44568-4_2.

[47] Luca Iocchi i in. „Distributed Coordination in Heterogeneous Multi-Robot Systems". W: *Autonomous Robots* 15 (sty. 2003), s. 155–168. DOI: 10.1023/A:1025589008533.

[48] Ole Høgh Jensen. „Mobile Processes in Bigraphs". Rozprawa doktorska. Cambridge University Computer Lab, paź. 2006. URL: https://www.cl.cam.ac.uk/archive/rm135/Jensen-monograph.pdf.

[49] Sanza T. Kazadi. „Swarm engineering". Phd thesis. California Institute of Technology, 2000. DOI: 10.7907/YR44-KF67.

[50] Charsles R.B. Wright Kenneth A. Ross. *Matematyka Dyskretna*. Wydawnictwo Naukowe PWN, 1996.

[51] Asif Khan, Bernhard Rinner i Andrea Cavallaro. „Cooperative Robots to Observe Moving Targets: Review". W: *IEEE Transactions on Cybernetics* 48.1 (2018), s. 187–198. DOI: 10.1109/TCYB.2016.2628161.

[52] Andreas Kolling i in. „Human Interaction With Robot Swarms: A Survey". W: *IEEE Transactions on Human-Machine Systems* 46.1 (2016), s. 9–26. DOI: 10.1109/THMS.2015.2480801.

[53] Bohdan Korzan. *Elementy teorii grafów i sieci - metody i zastosowania*. Wydawnictwa Naukowo-Techniczne, 1978.

[54] Jean Krivine, Robin Milner i Angelo Troina. „Stochastic Bigraphs". W: *Electronic Notes in Theoretical Computer Science* 218 (2008). Proceedings of the 24th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXIV), s. 73–96. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2008.10.006.

[55] Nuno Lau i in. „Multi-robot team coordination through roles, positionings and coordinated procedures". W: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, s. 5841–5848. DOI: 10.1109/IROS.2009.5354286.

[56] Antoine Ligot i in. „Automatic modular design of robot swarms using behavior trees as a control architecture". W: *PeerJ Computer Science* 6 (2020). DOI: 10.7717/peerj-cs.314.

[57] Lapierre Lionel. „Underwater Robots Part I: Current Systems and Problem Pose". W: *Mobile Robots*. Red. Aleksandar Lazinica. Rijeka: IntechOpen, 2006. Rozd. 16. DOI: 10.5772/4697.

[58] Alessio Mansutti, Marino Miculan i Marco Peressotti. „Multi-agent Systems Design and Prototyping with Bigraphical Reactive Systems". W: *Distributed Applications and Interoperable Systems*. Red. Kostas Magoutis i Peter Pietzuch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, s. 201–208. ISBN: 978-3-662-43352-2. DOI: 10.1007/978-3-662-43352-2_16.

[59] Maja J. Matarić. „From Local Interactions to Collective Intelligence". W: *The Biology and Technology of Intelligent Autonomous Agents*. Red. Luc Steels. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, s. 275–295. ISBN: 978-3-642-79629-6. DOI: 10.1007/978-3-642-79629-6_11.

[60] Brendan D. McKay i Adolfo Piperno. „Practical graph isomorphism, II". W: *Journal of Symbolic Computation* 60 (2014), s. 94–112. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2013.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0747717113001193.

[61] Hamid Menouar i in. „UAV-Enabled Intelligent Transportation Systems for the Smart City: Applications and Challenges". W: *IEEE Communications Magazine* 55 (mar. 2017), s. 22–28. DOI: 10.1109/MCOM.2017.1600238CM.

[62] Robin Milner. *The Space and Motion of Communicating Agents*. T. 20. Cambridge University Press, sty. 2009. ISBN: 978-0-521-73833-0. DOI: 10.1017/CBO9780511626661.

[63] Yogeswaran Mohan i S. G. Ponnambalam. „An extensive review of research in swarm robotics". W: *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*. 2009, s. 140–145. DOI: 10.1109/NABIC.2009.5393617.

[64] Amartya Mukherjee i in. „Flying Ad hoc Networks: A Comprehensive Survey". W: sty. 2018, s. 569–580. ISBN: 978-981-10-7562-9. DOI: 10.1007/978-981-10-7563-6_59.

[65] Rodrigo Munguia i in. „Design and Development of Aerial Robotic Systems for Sampling Operations in Industrial Environment". W: *Aerial Robots*. Red. Omar Dario Lopez Mejia i Jaime Alberto Escobar Gomez. Rijeka: IntechOpen, 2017. Rozd. 6. DOI: 10.5772/intechopen.70005.

[66] Nadia Nedjah i Luneque Silva Junior. „Review of methodologies and tasks in swarm robotics towards standardization". W: *Swarm and Evolutionary Computation* (2019). ISSN: 2210-6502. DOI: 10.1016/j.swevo.2019.100565.

[67] Lynne E. Parker, Daniela Rus i Gaurav S. Sukhatme. „Multiple Mobile Robot Systems". W: *Springer Handbook of Robotics*. Red. Bruno Siciliano i Oussama Khatib. Cham: Springer International Publishing, 2016, s. 1335–1384. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_53.

[68] Eloi Pereira i in. „Modeling and controlling the structure of heterogeneous mobile robotic systems: A bigactor approach". W: *2013 IEEE International Systems Conference (SysCon)*. 2013, s. 442–447. DOI: 10.1109/SysCon.2013.6549920.

[69] Helena Rasiowa. *Wstęp do matematyki współczesnej*. Wydawnictwo Naukowe PWN, 2012.

[70] Yara Rizk, Mariette Awad i Edward W. Tunstel. „Cooperative Heterogeneous Multi-Robot Systems: A Survey". W: *ACM Comput. Surv.* 52.2 (2019). ISSN: 0360-0300. DOI: 10.1145/3303848.

[71] Juan Jesús Roldán-Gómez, Eduardo González-Gironda i Antonio Barrientos. „A Survey on Robotic Technologies for Forest Firefighting: Applying Drone Swarms to Improve Firefighters' Efficiency and Safety". W: *Applied*

Sciences 11.1 (2021). ISSN: 2076-3417. DOI: 10.3390/app11010363. URL: https://www.mdpi.com/2076-3417/11/1/363.

[72] Stuart Russell i Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Pearson Education, 2016. ISBN: 9781292153964.

[73] Erol Şahin. „Swarm Robotics: From Sources of Inspiration to Domains of Application". W: *Swarm Robotics*. Red. Erol Şahin i William M. Spears. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 10–20. ISBN: 978-3-540-30552-1. DOI: 10.1007/978-3-540-30552-1_2.

[74] Zbigniew Semadeni. *Wstęp do teorii kategorii i funktorów*. Wyd. 2 rozszerz. Biblioteka Matematyczna t. 45. Warszawa: Państwowe Wydawnictwo Naukowe, 1978.

[75] Michele Sevegnani. *Bigraph tools*. URL: https://bitbucket.org/uog-bigraph/bigraph-tools.

[76] Michele Sevegnani i Muffy Calder. „BigraphER: Rewriting and Analysis Engine for Bigraphs". W: *Computer Aided Verification*. Red. Swarat Chaudhuri i Azadeh Farzan. Cham: Springer International Publishing, 2016, s. 494–501. ISBN: 978-3-319-41540-6. DOI: 10.1007/978-3-319-41540-6_27.

[77] Michele Sevegnani i Muffy Calder. „Bigraphs with sharing". W: *Theoretical Computer Science* 577.0 (2015), s. 43–73. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2015.02.011.

[78] Hazim Shakhatreh i in. „Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges". W: *IEEE Access* 7 (2019), s. 48572–48634. DOI: 10.1109/ACCESS.2019.2909530.

[79] Thomas B. Sheridan i William L. Verplank. „Human and Computer Control of Undersea Teleoperators". W: 1978.

[80] Marir Souad, Belala Faiza i Hameurlain Nabil. „Formal Modeling IoT Systems on the Basis of BiAgents and Maude". W: *2020 International Conference on Advanced Aspects of Software Engineering (ICAASE)*. 2020, s. 1–7. DOI: 10.1109/ICAASE51408.2020.9380126.

[81] William Spears i in. „Distributed, Physics-Based Control of Swarms of Vehicles". W: *Auton. Robots* 2 (czer. 2004). DOI: 10.1023/B:AURO.0000033970.96785.f2.

[82] Daniele Ventura i in. „Unmanned Aerial Systems (UASs) for Environmental Monitoring: A Review with Applications in Coastal Habitats". W: *Aerial Robots*. Red. Omar Dario Lopez Mejia i Jaime Alberto Escobar Gomez. Rijeka: IntechOpen, 2017. Rozd. 8. DOI: 10.5772/intechopen.69598.

[83] Nikos Vlassis. „A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence". W: *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2007). DOI: 10.2200/S00091ED1V01Y200705AIM002.

[84] Alan Winfield i Julien Nembrini. „Safety in Numbers: Fault Tolerance in Robot Swarms". W: *International Journal of Modelling Identification and Control* 1 (sty. 2006), s. 30–37. DOI: 10.1504/IJMIC.2006.008645.

[85] Zhu Xiaoning. „Analysis of military application of UAV swarm technology". W: *2020 3rd International Conference on Unmanned Systems (ICUS)*. 2020, s. 1200–1204. DOI: 10.1109/ICUS50048.2020.9274974.

[86] Lin Zhang i in. „Decentralized Control of Multi-Robot System in Cooperative Object Transportation Using Deep Reinforcement Learning". W: *IEEE Access* 8 (2020), s. 184109–184119. DOI: 10.1109/ACCESS.2020.3025287.

[87] Xiaonan Zhang, Pengcheng Luo i Xinwu Hu. „Defense Success Rate Evaluation for UAV Swarm Defense System". W: *Proceedings of the 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*. ISMSI '18. Phuket, Thailand: Association for Computing Machinery, 2018, s. 127–132. ISBN: 9781450364126. DOI: 10.1145/3206185.3206188.

[88] Donghui Zhao i in. „Multiple Welfare-Robots Architecture of Smart House for Security Assistance of Daily Activities". W: *2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. 2018, s. 595–601. DOI: 10.1109/IISR.2018.8535715.

[89] Yongkun Zhou, Bin Rao i Wei Wang. „UAV Swarm Intelligence: Recent Advances and Future Trends". W: *IEEE Access* 8 (2020), s. 183856–183878. DOI: 10.1109/ACCESS.2020.3028865.

Spis rysunków

2.1 Przykład dwugrafu z podziałem na składowe. 26
2.2 Przykład operacji kompozycji dwugrafów. 32
2.3 Przykład operacji zestawienia dwóch dwugrafów. 34
2.4 Dwugraf wejściowy do przykładu wprowadzającego 35
2.5 Przykład różnych wystąpień tego samego wzorca dwugrafowego . 37
2.6 Przykład wystąpienia wzorca dwugrafowego 40
2.7 Zbiór $\mathcal{B}$ przykładowego TBRS 44
2.8 Graficzna reprezentacja przejść TTS zdefiniowanego w tabeli 2.2. . 47

3.1 Przestrzeń stanów powstała z TTS z tabeli 2.2 54
3.2 Harmonogram działań dla marszruty $0 \xrightarrow{f_1} 1 \xrightarrow{f_4} 2$ w przestrzeni stanów z rysunku 3.1 62
3.3 Harmonogram działań dla marszruty $0 \xrightarrow{f_3} 2$ w przestrzeni stanów z rysunku 3.1 62
3.4 Całościowy obraz opracowanej metodyki - czynności wykonywane przez projektanta. 64
3.5 Całościowy obraz opracowanej metodyki - czynności wykonywane przez oprogramowanie. 65

4.1 Diagram zależności pomiędzy etapami weryfikacji modelu MAS. . 72
4.2 Schemat blokowy 4 etapu algorytmu weryfikacji modelu MAS. . . 77
4.3 Schemat blokowy 3 etapu algorytmu weryfikacji modelu MAS. . . 83
4.4 Schemat blokowy etapu 2 algorytmu weryfikacji modelu MAS. . . 88
4.5 Schemat blokowy etapu 1 algorytmu weryfikacji modelu MAS. . . 94

5.1 Stan początkowy dla przykładu współpracy pomiędzy dronami . . 96
5.2 Zbiór $\mathcal{B}$ w przykładzie 5.1 98

5.3 Przestrzeń stanów dla przykładu 5.1 - zadania wymagającego współpracy między dronami. 100
5.4 Harmonogram działań dla przykładu 5.1 101
5.5 Stan początkowy dla przykładu weryfikacji poprawnego modelu. . 102
5.6 Zbiór *Agt* dla TBRS z przykładu weryfikacji poprawności modelu. 103
5.7 Harmonogram działań dla przykładu 5.2 111
5.8 Stan początkowy dla przykładu weryfikacji niepoprawnego modelu. 112
5.9 Zbiór *Agt* dla TBRS z przykładu weryfikacji nieprawidłowego modelu. 113
5.10 Przestrzeń stanów z przykładu wykrycia błędów w modelu 114
5.11 Pierwsza postać stanu początkowego zadania z przykładu 5.4.1 . . 119
5.12 Druga postać stanu początkowego zadania, na potrzeby przykładu skalowania modelu w poziomie. 120
5.13 Stan początkowy systemu do przykładu skalowania projektu w pionie 122
5.14 Stan docelowy systemu po wykonaniu pierwszego zadania, w przykładzie 5.4.2. 126
5.15 Harmonogram działań dla pierwszego etapu zadania z przykładu 5.4.2 127
5.16 Harmonogram działań dla drugiego etapu zadania z przykładu 5.4.2 127
5.17 Harmonogram działań dla dronów realizujących obydwa etapy zadania, z przykładu skalowania w pionie 128
5.18 Stan początkowy zadania dla przykładu skalowania projektu w pionie, zobrazowany przy użyciu narzędzia [27]. 128
5.19 Stan docelowy systemu po wykonaniu pierwszego zadania, w przykładzie skalowania projektowanego zachowania w pionie, zobrazowany przy użyciu symulatora [27]. 129
5.20 Jeden z stanów scenariusza po chwili 3, ale przed chwilą 4, zobrazowany przy użyciu symulatora [27]. 129
5.21 Harmonogram działań dla przykładu 5.5, powstały z polityki zachowania o długości 15 131
5.22 Harmonogram działań dla przykładu 5.5, powstały z polityki zachowania o długości 18 132

5.23 Wzorce dwugrafowe dla przykładu weryfikacji wymagań pozafunkcjonalnych 132
5.24 Porównanie szybkości bibliotek *bigraph* i *tracking_ bigraph* 138
5.25 Przykład przekształcenia dwugrafu w kolorowany graf skierowany 142

B.1 Przykładowy element relacji reakcji przekształcany w przejście w TTS 183
B.2 Reguła reakcji, w oparciu o którą zdefiniowano relacje reakcji z rysunku B.1 183
B.3 Wystąpienia redeksu z rysunku B.2 w dwugrafie B_1 z rysunku B.1 184
B.4 Rysunek pomocniczy do załącznika B 186

Spis tabel

2.1 Zbiór reguł reakcji dla przykładowego TBRS 44
2.2 Przykład śledzącego systemu przejść 47

3.1 Definicje funkcji postępu misji przestrzeni stanów z rysunku 3.1 . 55

4.1 Parametry wejściowe 4 etapu 76
4.2 Wynik 4 etapu algorytmu do weryfikacji modelu. 76
4.3 Przykładowy przebieg 4 etapu algorytmu weryfikacji modeli MAS 79
4.4 Parametry wejściowe 3 etapu algorytmu do weryfikacji modelu. . . 82
4.5 Zmienne pomocnicze używane w trakcie wykonywania etapu 3. . . 84
4.6 Wynik 3 etapu algorytmu do weryfikacji modelu. 84
4.7 Parametry wejściowe algorytmu etapu 2. 87
4.8 Zmienne pomocnicze używane w trakcie wykonywania etapu 2 wraz z jego wynikiem. 89
4.9 Przykładowy wynik etapu 2 91
4.10 Parametry wejściowe algorytmu etapu 1. 92
4.11 Zmienne pomocnicze algorytmu etapu 1. 93
4.12 Wyniki 1 etapu algorytmu do weryfikacji modelu 93

5.1 Reguły reakcji dla przykładu 5.1 97
5.2 Śledzący system przejść dla przykładu 5.1 97
5.3 Funkcje postępu misji dla przykładu 5.1 99
5.4 Reguły reakcji dla przykładu weryfikacji poprawności modelu . . . 102
5.5 Śledzący system przejść dla przykładu weryfikacji modelu 103
5.6 Funkcje postępu misji dla przykładu weryfikacji modelu 104
5.7 Przebieg weryfikacji poprawności prawidłowego modelu MAS . . . 105
5.8 Reguły reakcji w przykładzie weryfikacji nieprawidłowego modelu 113

5.9 TTS dla przykładu weryfikacji nieprawidłowego modelu 113
5.10 Funkcje postępu misji dla przykładu weryfikacji nieprawidłowego modelu 114
5.11 Przebieg weryfikacji przykładowego nieprawidłowego modelu. . . . 115
5.12 Interpretacja sterowników dla przykładu skalowania w poziomie . 119
5.13 Reguły reakcji w przykładzie weryfikacji wymagań pozafunkcjonalnych 120
5.14 Reguły reakcji dla pierwszego zadania z przykładu 5.4.2 125
5.15 Reguły reakcji dla drugiego zadania z przykładu 5.4.2 126
5.16 Parametry wariantów zadania użytego do porównania szybkości bibliotek *bigraph* i *tracking_ bigraph* 136

F.1 Funkcje postępu misji do analizy przebiegu etapu 2 206
F.2 Przebieg etapu 2 algorytmu weryfikacji poprawności modelu MAS dla przykładu z rozdziału 5.2. 206

Załącznik A

Weryfikacja wyników eksperymentów

Tematem tego załącznika będzie zademonstrowanie, w jaki sposób można wykorzystać oprogramowanie [22, 26, 23, 24] do konstrukcji oraz weryfikacji polityki zachowania, realizującej zadanie omówione w rozdziale 3. Omawiane tutaj programy zostały przygotowane w ramach artykułów [29, 28]. Opisane czynności mogą również posłużyć do opracowania i weryfikacji polityki zachowania dla nowych zadań.

Biblioteka programowa [22] służąca do konstrukcji śledzących systemów przejść, zbiór programów [26] do normalizacji i przekształcenia TTS w przestrzeń stanów, a także oprogramowanie do weryfikacji polityki zachowania [24] zostały opracowanie w języku *OCaml*. Narzędzie [23] do wyznaczenia polityki zachowania w przestrzeni stanów zostało przygotowane w języku *F#*. Czynności omawiane w tym załączniku są opisywane z założeniem, że środowisko, w którym będą wykonywane jest przygotowane w następujący sposób:

- Menadżer pakietów dla języka OCaml - OPAM umożliwia kompilowanie programów wykorzystujących biblioteki: `bigraph`[1] oraz `tracking_bigraph`. Minimalna akceptowalna wersja kompilatora OCaml to 4.12.0.
- Zainstalowano platformę .NET w wersji co najmniej 5.0 .

Biblioteka programowa [22] umożliwia konstrukcję śledzącego systemu przejść,

[1]Dostępna pod adresem: `https://uog-bigraph.bitbucket.io/`

w oparciu o stan początkowy systemu oraz zbiór reguł reakcji. Stan początkowy dla tego zadania w postaci graficznej został przedstawiony na rysunku 2.4. Reguły reakcji dla tego zadania zamieszczono w tabeli 2.1. Obydwa te elementy muszą zostać przedstawione przy użyciu struktur udostępnionych przez biblioteki `bigraph` oraz `tracking_bigraph`.

W celu zdefiniowania dwugrafu należy określić funkcję przypisującą identyfikatorom wierzchołków sterowniki, wraz z liczbą portów jaką każdy wierzchołek posiada. Ponadto trzeba zdefiniować graf rozmieszczenia i połączeń. Wszystkie te trzy elementy mają postać tekstową i są konstruowane w następujący sposób:

- Funkcja definiująca wierzchołki dwugrafu zapisywana jest jako zbiór par, w których argumentem jest identyfikator wierzchołka, a wartością jest typ sterownika wraz z liczbą portów. Dla dwugrafu z rysunku 2.1 omawiana funkcja będzie zapisywana jako:

  ```
  {(0, A:1),(1, A:1),(2, C:1),(3, C:1),(4, B:2)}
  ```

- Graf rozmieszczenia jest definiowany przy użyciu struktury przypominającej macierz incydencji poprzedzonej trzema wartościami. Pierwsza oznacza liczbę źródeł, druga liczbę wierzchołków, a trzecia liczbę parceli. Kolejne wiersze zawierają zbiór składający się z 0 lub 1, które oznaczają, czy element reprezentowany przez kolumnę jest następnikiem elementu reprezentowanego przez wiersz. Wiersze reprezentują kolejne źródła i wierzchołki, a kolumny wierzchołki i parcele. Graf rozmieszczenia z rysunku 2.1b będzie przedstawiony przy użyciu powyższej notacji w następujący sposób:

  ```
  2 5 3
  10000000
  01000000
  00110000
  00001000
  00000100
  00000010
  00000001
  ```

- Graf połączeń jest definiowany jako zbiór trójek uporządkowanych, których elementami są zbiory połączeń wewnętrznych oraz zewnętrznych wraz ze zbiorem par, których elementami są funkcje przypisujące wierzchołkom krotność incydencji z hiperkrawędziami. Graf połączeń z rysunku 2.1c może być przedstawiony w następujący sposób, przy użyciu powyżej opisanej notacji:

```
({}, {on1}, {(0, 1), (1, 1)})
({}, {}, {(2, 1), (3, 1)})
({in1}, {}, {(4, 1)})
({}, {}, {(4, 1)})
```

Biblioteka `tracking_bigraph` umożliwia konstrukcję reguł reakcji jako pary dwugrafów oraz dwóch funkcji. Pierwsza z funkcji przypisuje parcele w reaktum parcelom w redeksie. Umożliwia to konstruowanie reguł reakcji, które nie są *liniowe* (według definicji z rozdziału 2.4). W związku z tym, że w tej pracy rozpatrywane są tylko liniowe reguły reakcji to funkcja ta będzie zawsze tożsamością[2] i_n, gdzie n to liczba parceli w redeksie. Drugą funkcją jest funkcja śledząca zdefiniowana w 2.4.

Fragment kodu w języku OCaml, który definiuje dwugraf będący stanem początkowym systemu przedstawiony na rysunku 2.4 oraz elementy niezbędne do konstrukcji reguł reakcji z tabeli 2.1 przedstawiono poniżej.

```
1  (* Konstrukcja TTS - czesc 1 - jezyk OCaml *)
2  let s0_string =
3  "
4  {(0, A:1),(1, U:0),(2, U:0),(3, B:1)}
5  0 4 0
6  0110
7  0000
8  0000
9  0000
10 ({}, {}, {(0, 1), (3, 1)})
11 "
```

[2]Zarówno w bibliotece `bigraph`, jak i `tracking_bigraph` funkcje konstruujące reguły reakcji mogą przyjąć wbudowaną w OCaml wartość `None` w sytuacji, gdy funkcja ta jest tożsamością. W dalszej części pracy będzie to wykorzystywane do skrócenia zapisu.

```
let r1_redex_string =
"
{(0, A:1),(1, U:0),(2, B:1)}
0 3 2
01010
00000
00001
({}, {}, {(0, 1), (2, 1)})
"
let r1_reactum_string =
"
{(0, A:1),(1, U:0),(2, B:1)}
0 3 2
00010
00000
01001
({}, {}, {(0, 1), (2, 1)})
"
let r2_redex_string =
"
{(0, A:1),(1, U:0),(2, U:0),(3, B:1)}
0 4 0
0110
0000
0000
0000
({}, {}, {(0, 1), (3, 1)})
"
let r2_reactum_string =
"
{(0, A:1),(1, U:0),(2, U:0),(3, B:1)}
0 4 0
0000
0000
0000
0110
({}, {}, {(0, 1), (3, 1)})
"
```

Korzystając z funkcji w bibliotece `tracking_bigraph` należy przekształcić powyższe elementy w formie tekstowej na dwugraf oraz reguły reakcji. Następnie można wykorzystać przekształcone elementy do konstrukcji śledzącego systemu przejść. Interfejs udostępniany w ramach opracowanej biblioteki wymaga określenia stanu początkowego systemu, listy reguł reakcji oraz liczby oznaczającej maksymalną odległość od stanu początkowego. Ostatni parametr może być interpretowany jako maksymalna liczba przejść (w TTS) od stanu początkowego do dowolnego stanu. Wynikiem są dwa pliki CSV (ang. *comma separated value*), których zawartość jest następująca:

- Kolejne wiersze pierwszego pliku zawierają indeks stanu systemu oraz jego reprezentacje w formie tekstowej opisanej wcześniej. Jest bardzo ważnym, aby podkreślić, że nie ma w tym pliku dwóch stanów (dwugrafów) równoważnych z dokładnością do podstawy.
- Drugi plik CSV zawiera przejścia w skonstruowanym TTS. Kolejne kolumny w każdym wierszu oznaczają:
  - Indeks stanu początkowego;
  - Indeks stanu wynikowego;
  - Etykieta reakcji;
  - Funkcja uczestnictwa;
  - Funkcja pozostałości;
  - Reprezentacja tekstowa stanu wynikowego.

Fragment kodu w języku OCaml, który przekształca wcześniejsze elementy z reprezentacji tekstowej oraz generuje TTS w postaci dwóch wyżej opisanych plików został zamieszczony poniżej.

```
1 (* Konstrukcja TTS - czesc 2 - jezyk OCaml *)
2 open Bigraph
3 open Tracking_bigraph
4
5 let s0 = Big.of_string s0_string
6 let r1_redex = Big.of_string r1_redex_string
7 let r1_reactum = Big.of_string r1_reactum_string
```

```
8  let r2_redex = Big.of_string r2_redex_string
9  let r2_reactum = Big.of_string r2_reactum_string
10 let r1_residue = Fun.of_list [(0,0);(1,1);(2,2)]
11 let r2_residue = Fun.of_list [(0,0);(1,1);(2,2);(3,3)]
12 let r1 = TBrs.parse_react "r1" ~lhs:r1_redex ~rhs:r1_reactum ~
    f_sm:None ~f_rnm:r1_residue
13 let r2 = TBrs.parse_react "r2" ~lhs:r2_redex ~rhs:r2_reactum ~
    f_sm:None ~f_rnm:r2_residue
14
15 let max_steps = 7
16
17 module Gen = TBrs.Make(Tools.DigraphTools);;
18
19 let _ = Gen.explore_ss_slim
20           ~trans_file_name:"transitions.csv"
21           ~states_file_name:"states.csv"
22           s0
23           [r1;r2]
24           max_steps
```

Konstruuje on TTS, w którym maksymalna liczba przejść od stanu początkowego do dowolnego innego stanu jest równa 7. Jego wynikiem są dwa pliki: *states.csv* oraz *transitions.csv*.

Mając TTS, będący wynikiem działania biblioteki `tracking_bigraph`, należy dokonać normalizacji zawartych w nim przejść. Wynik nieznormalizowany zawiera w ostatniej kolumnie pliku z przejściami dwurafy, które są równoważne z dokładnością do podstawy dwugrafom stanu wynikowego z pliku zawierającego wszystkie stany. Proces normalizacji ma na celu przekształcenie wszystkich dwugrafów, będących stanem wynikowym dowolnego przejścia, do postaci identycznej odpowiadającemu mu stanowi w pliku zawierającym wszystkie stany systemu wraz z indeksami. Wiąże się to z przekształceniem funkcji uczestnictwa oraz pozostałości dla każdego przejścia, w którym dwugraf w ostatniej kolumnie różni od odpowiadającemu mu dwugrafu w pliku ze stanami systemu.

W celu znormalizowania wcześniej opracowanego systemu przejść można użyć programu, udostępnionego w ramach zestawu narzędzi przekształcających TTS w SS [26]. Przyjmując, że wynikiem biblioteki `tracking_bigraph` są pliki *sta-*

tes.csv oraz *transitions.csv* znajdujące się w tym samym folderze co program `normalize_tts.exe`, będący częścią projektu [26], normalizacja przejść może zostać wykonana w następujący sposób:

```
$ ./normalize_tts.exe \
      states.csv \
      transitions.csv \
      normed_trans.csv \
      false
```

Wynikiem powyższej komendy będzie plik `normed_trans.csv`, zawierający znormalizowane przejścia z TTS.

Mając powyższy wynik można przystąpić do przekształcenia (znormalizowanego) systemu przejść w przestrzeń stanów. Przekształcenie to można wykonać również z użyciem programu opracowanego w ramach [26]. W tym celu potrzeba przygotować dwa pliki CSV, których zawartość będzie następująca:

- Plik zawierający listę reguł reakcji wraz z czasem wykonania czynności, które reprezentują. Jest to dwukolumnowy beznagłówkowy plik CSV, którego pierwsza kolumna zawiera nazwę reguły reakcji, a druga czas potrzebny na wykonanie powiązanej z nią czynności. Dla omawianego przykładu plik ten będzie określany jako *t2ts.csv*, a jego zawartość następująca:

```
r1,2
r2,1

```

- Plik zawierający listę sterowników, które są utożsamiane z obiektami zadania. Jest to plik, który w kolejnych wierszach zawiera sterownik wraz z liczbą portów. Dla omawianego przykładu plik ten, określany dalej jako *ctrls.csv*, będzie miał następującą postać:

```
U:0

```

Korzystając z programu `transoform_normed_tts`, będącego elementem projektu [26], można przekształcić znormalizowany system przejść w następujący sposób:

```
$ ./transform_normed_tts.exe \
      states.csv \
      normed_trans.csv \
      t2ts.csv \
      ctrs.csv \
      ss.csv
```

Wynikiem powyższego polecenia będzie plik *ss.csv* zawierający funkcje postępu misji, który może posłużyć do opracowania polityki zachowania.

Do konstrukcji polityki zachowania dla obiektów zadania przygotowano oprogramowanie[3] [23], które akceptuje przestrzeń stanów w postaci pliku CSV.

Program `SSPApp` będący fasadą projektu [23] umożliwia konstrukcję polityki zachowania na podstawie przestrzeni stanów, będącej wynikiem programu `transform_normed_tts` oraz dwukolumnowego pliku CSV. Pierwsza kolumna tego pliku zawiera indeks stanu docelowego, a druga komentarz, dlaczego dany stan jest satysfakcjonujący. Do poprawnego działania programu `SSPApp` wymagany jest plik konfiguracyjny w formacie JSON. Może być on przygotowany w pełni odręcznie lub poprzez zmodyfikowanie szablonu wygenerowanego przez sam program. W celu skonstruowania szablonu należy uruchomić program `SSPApp` w następujący sposób:

```
$ ./SSPApp exmp myConfigFile.json
```

gdzie *myConfigFile.json* to nazwa pliku, który będzie wynikiem działania programu. Zakładając, że pliki CSV zawierające funkcje postępu misji oraz listę akceptowalnych stanów docelowych znajdują się w tym samym folderze co program `SSPApp`, plik konfiguracyjny dla rozpatrywanego zadania będzie miał następującą zawartość:

```
{
  "numberOfAgents": 2,
  "numberOfStates": 3,
  "transitionFunctionsFile": "ss.csv",
  "destinationStatesFile": "dest_states.csv",
  "numberOfSteps": 4,
```

[3]Opracowano dwie wersje tego programu. Pierwsza (wcześniejsza) została wykonana w języku OCaml. Jednak ze względu na to, że język ten aktualnie nie umożliwia programowania wielowątkowego ze współdzieloną pamięcią, przygotowana została druga wersja w języku F# i to właśnie sposób użycia tej wersji będzie przedstawiony.

```
    "outputFilePrefix": "intro",
    "task": "search until",
    "resultStrategy": "all",
    "destinationStrategy": "first found",
    "forceNoIdling": false,
    "printSATs": true
}
```

Znaczenie kolejnych elementów pliku konfiguracyjnego jest następujące:

- `numberOfAgents` - liczba obiektów w zadaniu.
- `numberOfStates` - liczba stanów w systemie.
- `transitionFunctionsFile` - ścieżka względna do pliku z przestrzenią stanów w formacie CSV.
- `destinationStatesFile` - plik z listą indeksów stanów rozpatrywanych jako docelowe (w tym przypadku jest jeden taki stan o indeksie 2).
- `numberOfSteps` - maksymalna liczba iloczynów macierzy $\mathbb{M}_F \cdot \mathbb{M}_K$ wykonanych w celu znalezienia polityki zachowania.
- `outputFilePrefix` - prefiks plików będących wynikiem działania programu, które zawierają pojedynczą politykę zachowania.
- `task` - zadanie do wykonania przez program. Akceptowalne wartości to: `first found` (`count`) lub `search until` (`count`). Program wykonujący zadanie `first found` szuka najkrótszych marszrut prowadzących do satysfakcjonującego stanu, które jednocześnie nie są dłuższe niż `numberOfSteps`. Znalezione wyniki są zapisywane do plików CSV. W przypadku zadania `first found count` wynik nie jest zapisywany do pliku, a jedynie podawana jest liczba znalezionych marszrut. Program wykonujący zadanie `search until` szuka wszystkich marszrut prowadzących do satysfakcjonującego stanu, nie dłuższych niż `numberOfSteps` i zapisuje je do plików. Analogicznie `search until count` nie zapisuje wyników, a jedynie zwraca liczbę znalezionych polityk zachowania.

- `resultStrategy` - metoda ograniczająca liczbę wyników. Dostępne wartości tego parametru to: `first:N`, `all` lub `best:N`. Pierwsza metoda zwraca N pierwszych znalezionych wyników. Druga wartość tego parametru nie ogranicza w żaden sposób wyników i zwraca wszystkie znalezione marszruty pomiędzy stanem początkowym, a stanem docelowym. Ostatni sposób ograniczania liczby wyników polega na zwróceniu N najlepszych marszrut. W ramach biblioteki `SSPLib` jest wbudowana jedna funkcja, która umożliwia ocenę marszrut. Analizuje ona stopień zaangażowania obiektów zadania i tym lepiej ocenia marszrutę, im „równiej" zaangażowane są wszystkie obiekty.

- `destinationStrategy` - sposób wyboru stanu docelowego spośród wymienionych w pliku wskazanym przez `destinationStatesFile`. Akceptowalne wartości tego parametru to: `first found` oraz `random`. Wybór pierwszej wartości spowoduje, że program `SSPApp` będzie szukał marszrut od stanu 0 do stanu o indeksie wskazanym w pierwszym wierszu pliku `destinationStatesFile`. W przypadku, gdy wartością tego parametru jest `random`, poszukiwane są marszruty od stanu 0 do losowo wybranego stanu spośród wymienionych w pliku `destinationStatesFile`.

- `forceNoIdling` - parametr określający, czy w przypadku znalezienia polityk zachowania do wskazanego stanu zapomnieć o nich po zapisaniu. Ma to znaczenie tylko w przypadku szukania wszystkich polityk zachowania do zadanej długości (tj. gdy parametr `task` ma wartość `search until`). Dopuszczalne wartości to `true` lub `false`.

- `printSATs` - zmienna określająca, czy dla każdej znalezionej marszruty wypisać kolejne konfiguracje SaT, przechodząc od stanu 0 do stanu docelowego. Akceptowalne wartości to `true` lub `false`.

Aby uruchomić program `SPPApp` z plikiem konfiguracyjnym *policyConfig.json* znajdującym się w tym samym folderze należy użyć polecenia:

```
$ ./SSPApp gen policyConfig.json
```

Wynikiem będą trzy pliki CSV: `intro_0.csv`, `intro_1.csv`, `intro_2.csv`. Każdy

z nich zawiera kolejne funkcje postępu misji, które stanowią marszrutę w przestrzeni stanów, prowadzącą do satysfakcjonującego stanu.

Ostatnim etapem jest weryfikacja polityki zachowania. Proces ten można wykonać korzystając z programu `verify_visjs`, będącego częścią projektu [24]. Umożliwia on jednoczesną weryfikację modelu oraz przygotowanie danych, pozwalających zobrazować przebieg wykonania zadania przy użyciu [25]. Program `verify_visjs` potrzebuje do działania pliku konfiguracyjnego w formacie JSON, zawierającego opis scenariusza oraz kolory pomagające rozróżnić sterowniki. Dla omawianego przykładu wspomniany plik konfiguracyjny (określany dalej jako *verifConfig.json*) będzie miał następującą zawartość:

```
{
  "scenario_description": "Zadanie wprowadzajace do opracowanej
  metodyki.",
  "controls_2_color":[]
}
```

Ponadto program `verify_visjs` wymaga pliku CSV ze wszystkimi stanami systemu (rezultat działania biblioteki `tracking_bigraph`), znormalizowanych przejść (wynik programu `normalize_tts`), marszruty w formacie CSV (jeden z wyników `SSPApp`), liczby obiektów w zadaniu oraz prefiksu, który będą posiadały wszystkie pliki będące wynikiem działania programu. Wiedząc powyższe, w celu weryfikacji omawianego tutaj modelu należy uruchomić program `verify_visjs` w następujący sposób:

```
mkdir intro_content
$ ./verify_visjs.exe \
      verifConfig.json \
      states.csv \
      normed_trans.csv \
      walk.csv \
      2 \
      intro
```

Rezultatem działania tego programu będzie plik `intro_trs-ssp-output.json` oraz zbiór plików zapisany w folderze `intro_content`. Wspomniany plik wraz z folderem i jego zawartością służą jako dane wejściowe dla narzędzia [25]. Sposób jego obsługi nie będzie tutaj omówiony. W przypadku weryfikacji innego modelu,

która zakończy się niepowodzeniem, przedstawiony zostanie stosowny komunikat z informacją, co jest powodem niepoprawności modelu.

Załącznik B

Sposób przekształcenia TBRS w TTS

W tym załączniku zostanie przedstawiony przykładowy sposób, w jaki można przekształcić śledzący dwugrafowy system reakcyjny (TBRS) w śledzący system przejść (TTS). Przedstawiony tu proces ma na celu zademonstrowanie sposobu przejścia pomiędzy kolejnymi typami struktur matematycznych, a nie gotowy algorytm do zaimplementowania.

W celu przekształcenia TBRS w TTS potrzeba przekształcić elementy relacji reakcji zdefiniowanych nad zbiorem dwugrafów $\mathcal{B}$. Zbiór reguł reakcji, które mogą być „stosowane" do dwugrafów ze zbioru $\mathcal{B}$ będzie oznaczany $\mathcal{R}$.

Algorytm przekształcenia dowolnej reakcji

w zbiór przejść TTS jest następujący:

1. Znaleźć wszystkie nieizomorficzne wystąpienia redeksu reguły r_x w dwugrafie B_1. Każde z powyższych wystąpień redeksu r_x będzie kandydatem do przejścia w TTS, którego elementem *Apl* będzie para (B_1, r_x), a dwugrafem wynikowym będzie B_2. Dwugraf B_1 będzie określany dwugrafem wejściowym.

2. Funkcja przypisująca wierzchołki dwugrafu B_1 wierzchołkom redeksu r_x będzie stanowić funkcję *par* przejścia, skonstruowanego na podstawie tego wystąpienia redeksu w B_1.

3. Wystąpienie redeksu w B_1 jest zastępowane przez reaktum (ze zmienioną

podstawą). Wynikiem jest dwugraf B_2'. Jeżeli B_2' jest równoważny z dokładnością do podstawy dwugrafowi B_2, to konstruowane jest przejście w TTS. Jeżeli B_2' nie jest równoważny B_2, to przerywany jest proces konstrukcji przejścia na podstawie rozpatrywanego wystąpienia redeksu w B_1.

4. Funkcja przypisująca wierzchołki reaktum wierzchołkom wspomnianego zastąpienia redeksu będzie oznaczana przez $\rho_{B_2' \to react}$.

5. Przekształcenie podstawy dwugrafu B_2 w podstawę dwugrafu B_2' będzie oznaczane przez $\rho_{B_2 \to B_2'}$.

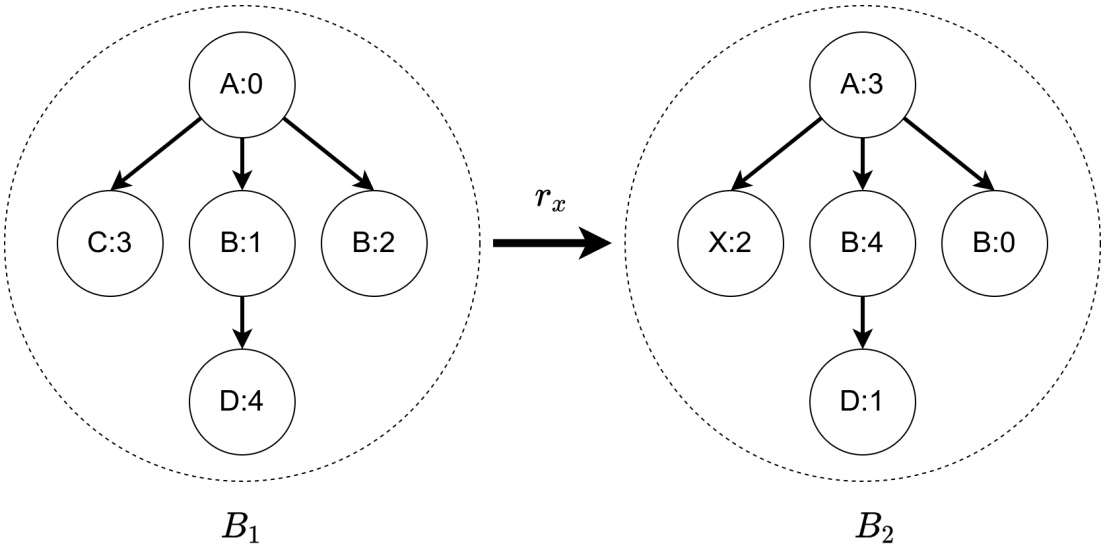
6. Korzystając z powyższych elementów oraz funkcji τ, będącej elementem śledzącej reguły reakcji r_x, funkcję res można zdefiniować jako:

$$res(v) = \begin{cases} par \circ \tau \circ \dot{\rho}(v) & \text{jeżeli funkcja } \tau \circ \dot{\rho}(v) \text{ jest określona} \\ \rho_{B_2 \to B_2'}(v) & \text{w przeciwnym przypadku} \end{cases}$$

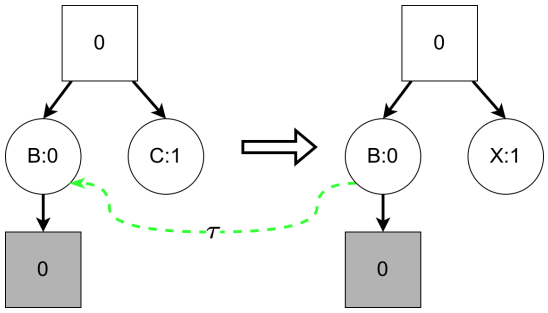
   gdzie $\dot{\rho} = \rho_{B_2' \to react} \circ \rho_{B_2 \to B_2'}$. Należy tutaj podkreślić, że funkcja res może być nieokreślona dla części wierzchołków B_2. Będzie tak w sytuacji, gdy τ jest funkcją częściową.

7. Ostatnim elementem jest konstrukcja przejścia w TTS wykorzystując opracowane powyżej funkcje oraz wiedząc, że przejście to jest z dwugrafu B_1 do B_2, z wykorzystaniem reguły reakcji r_x.

Jako przykład rozpatrzymy pojedynczą parę dwugrafów, przedstawionych na rysunku B.1, będących w relacji reakcji r_x. Reguła reakcji r_x została przedstawiona na rysunku B.2. Funkcja śledząca τ dla tej reguły reakcji ma postać $\tau = \{(0,0)\}$ i została zaznaczona zieloną przerywaną strzałką na rysunku B.2. Jest ona funkcją częściową, gdyż nie przypisuje wszystkim elementom dziedziny (wierzchołkom reaktum) wartości (wierzchołka z redeksu). Ponadto nie jest ona suriekcją, ponieważ nie wszystkie wierzchołki redeksu są wartościami funkcji. W tym przypadku wierzchołek o identyfikatorze 1 nie jest elementem przeciwdziedziny. Taką regułę reakcji można interpretować jako czynność, w której element oznaczony przez wierzchołek B, biorący udział w czynności istnieje po jej wykonaniu, podczas gdy element reprezentowany przez wierzchołek typu C prze-



Rysunek B.1: Element relacji reakcji na potrzeby przykładu przekształcenia TBRS w TTS. Elementy w okręgach narysowanych przerywaną linią stanowią dwugrafy, będące w relacji reakcji r_x.



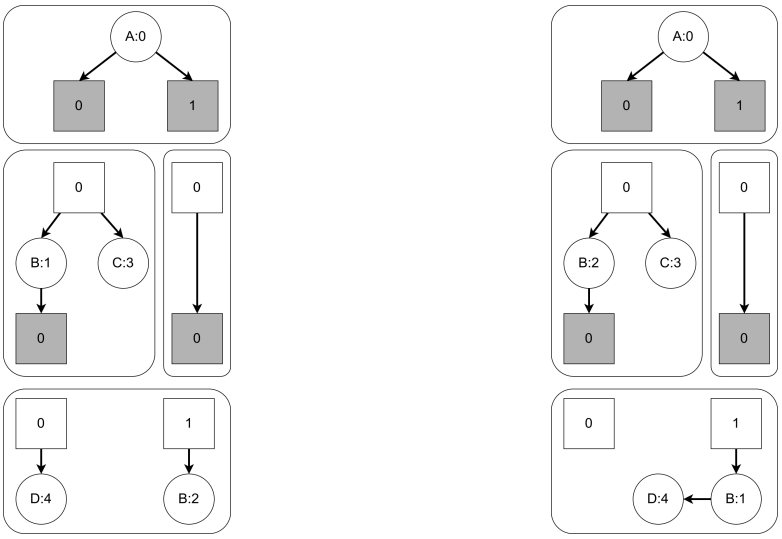
Rysunek B.2: Reguła reakcji r_x na potrzeby przykładu przekształcenia TBRS w TTS. Przerywana zielona strzałka oznacza funkcję śledzącą τ.

staje istnieć (przeciwdziedzina funkcji τ nie zawiera identyfikatora wierzchołka C). Ponadto w wyniku wykonania czynności r_x powstaje nowy element, oznaczony wierzchołkiem typu X (funkcja τ jest nieokreślona dla identyfikatora wierzchołka X).

Zgodnie z procedurą opisaną na początku tego rozdziału, kolejne etapy przekształcania elementu relacji reakcji r_x z rysunku B.1 są następujące:

1. Szukane są wszystkie nieizomorficzne wystąpienia redeksu w dwugrafie wejściowym B_1. Dla omawianego przykładu będą dwa takie wystąpienia. Obydwa zostały przedstawione na rysunku B.3.

2. Funkcja uczestnictwa dla „lewego” wystąpienia redeksu r_x ma postać $par = \{(0,1),(1,3)\}$. Z kolei dla wystąpienia po prawej stronie rysunku B.3 funk-



Rysunek B.3: Wystąpienia redeksu reguły r_x w dwugrafie B_1.

cją tą będzie $par = \{(0,2),(1,3)\}$.

3. Zastąpienie tylko „lewego" wystąpienia redeksu daje w wyniku dwugraf, równoważny z dokładnością do podstawy dwugrafowi B_2. W związku z tym tylko to wystąpienie redeksu będzie przekształcone w przejście w TTS.

4. Przypuśćmy, że w ramach procesu zmiany podstawy[1] reaktum, funkcja przypisująca wierzchołki reaktum reguły r_x wierzchołkom dwugrafu, który zastąpił redeks ma postać: $\rho_{B'_2 \to react} = \{(3,0),(1,1)\}$.

5. Przekształcenie podstawy[2] dwugrafu B_2 w podstawę dwugrafu B'_2 będzie bijekcją:

$$\rho_{B_2 \to B'_2} = \{(3,0),(4,3),(2,1),(1,4),(0,2)\}$$

6. Ostatnim etapem jest wyznaczenie funkcji res, dla przejścia konstruowanego na podstawie rozpatrywanego wystąpienia redeksu w dwugrafie B_1. Dla tego przypadku funkcja ta będzie miała następującą postać:

$$res = \{(3,0),(4,1),(1,4),(0,2)\}$$

[1]Celem tego przekształcenia jest, aby podstawa reaktum w dwugrafie B'_2 była rozłączna z podstawami pozostałych dwugafów, wchodzących w skład jego dekompozycji.

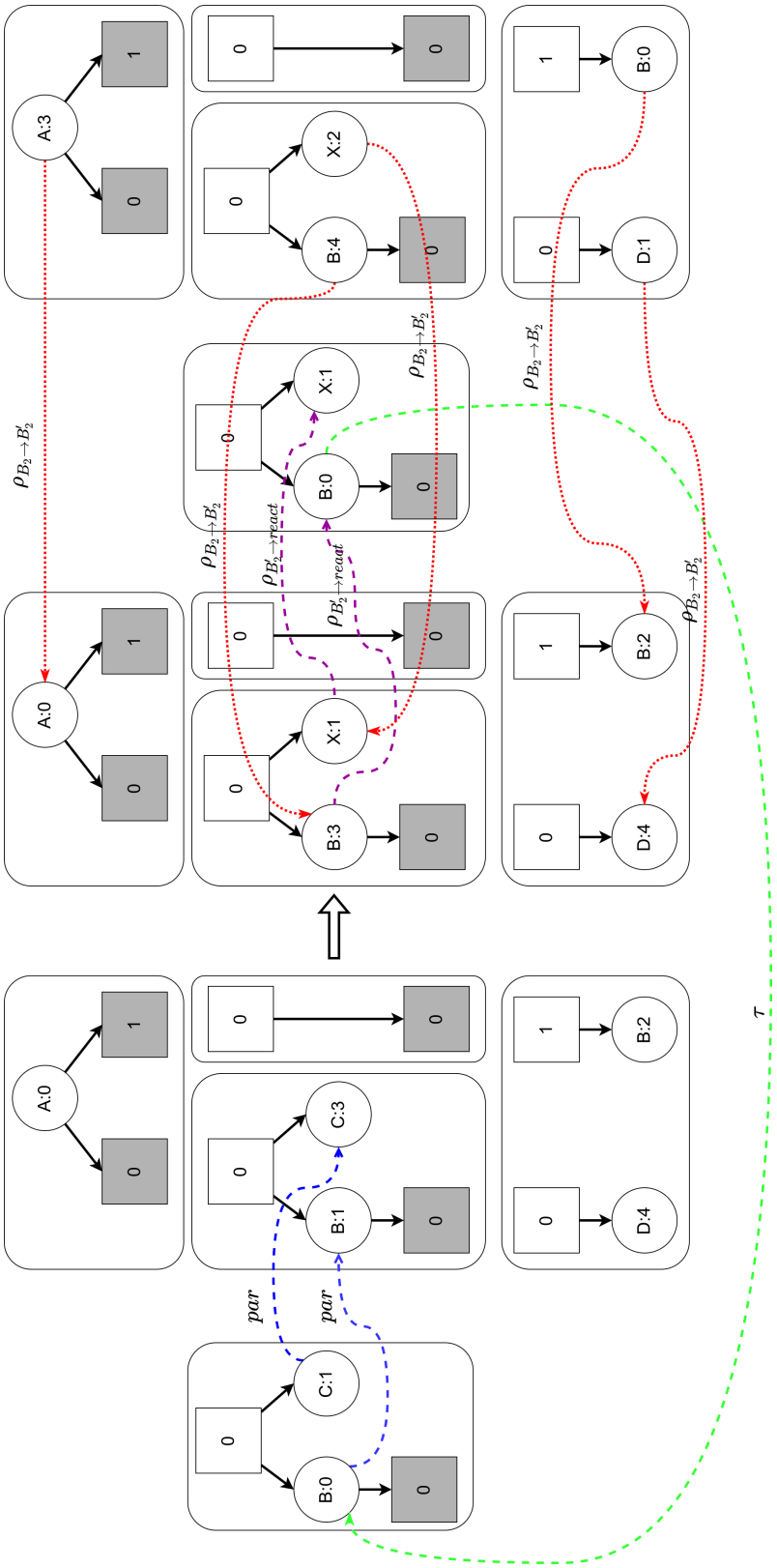
[2]Przedstawione tutaj przekształcenie podstawy ogranicza się do podstawy grafu rozmieszczenia, ponieważ graf połączeń nie zawiera żadnych hiperkrawędzi ani połączeń zewnętrznych/wewnętrznych. Przekształcenie podstawy zgodnie z definicją w 2.3 zawiera zarówno przekształcenie grafu rozmieszczenia jak i połączenia.

Wierzchołek o identyfikatorze 2 w dwugrafie B_2 nie jest pozostałością po żadnym wierzchołku w dwugrafie B_1. Jest tak dlatego, że funkcja τ reguły r_x jest częściowa i nie przypisuje wierzchołkowi X żadnego wierzchołka z redeksu. Innymi słowy, dla argumentu 1 funkcja τ jest nieokreślona. Można również zauważyć, że element reprezentowany przez wierzchołek o identyfikatorze 3 z dwugrafu B_1 nie występuje w dwugrafie B_2. Nie ma bowiem w przeciwdziedzinie funkcji res wartości 3.

7. Przejście w wynikowym TTS, skonstruowane na podstawie rozpatrywanego wystąpienia redeksu w B_1 będzie następujące:

Apl	Agt	Par	Res
$\langle B_1, r_x \rangle$	B_2	$\{(0,2),(1,3)\}$	$\{(3,0),(4,1),(1,4),(0,2)\}$

Omawiane powyżej etapy przekształcenia pojedynczego elementu relacji reakcji w zbiór (w tym przypadku jednoelementowy) przejść w TTS można prześledzić na rysunku B.4. Zostały na nim zamieszczone wszystkie powyższe funkcje elementarne. Proces ten można opisać w skrócie jako próbę, jak „najgłębszego" złożenia funkcji $par \circ tau \circ \rho_{B_2' \to react} \circ \rho_{B_2 \to B_2'}$. Funkcja res jest wynikiem złożenia kilku funkcji, przez co nie została zamieszczona.



Rysunek B.4: Rysunek pomocniczy do załącznika B. Zawiera wszystkie elementy, niezbędne do wyznaczenia funkcji res.

Załącznik C

Sposób przekształcenia TTS w SS

W tym załączniku zostanie przedstawiony sposób, w jaki można przekształcić śledzący system przejść, zdefiniowany w 2.5, w przestrzeń stanów zdefiniowaną w 3.2. Podobnie jak w poprzednim rozdziale algorytm tutaj przedstawiony nie ma postaci gotowej do implementacji, a jedynie ma na celu demonstrację sposobu przekształcenia wspomnianych wyżej struktur.

Procedura przekształcenia TTS w SS zakłada następujące kroki:

1. Określana jest liczba obiektów w zadaniu, oznaczana przez n_o.

2. Definiowany jest zbiór typów wierzchołków (sterowników w terminologii dwugrafów), które reprezentują obiekty zadania. Zbiór ten oznaczany będzie przez $\mathbb{O}$.

3. Dwugrafy będące elementami zbioru Agt przekształcane są w stany systemu (elementy zbioru S). Wynikiem jest iniekcja, przypisująca dwugrafom liczby naturalne $states : Agt \to \mathbb{N}$.

4. Każdej etykiecie ze zbioru Lab przypisywany jest czas wykonywania czynności, którą reprezentuje. Funkcja przypisująca czas do etykiet czynności będzie oznaczana przez $time : Lab \to \mathbb{N}$.

5. Każde przejście w przekształcanym TTS jest transformowane w funkcję postępu misji, według następującego schematu:

   (a) Definiowana jest etykieta jednoznacznie identyfikująca transformowane przejście. Etykieta ta będzie oznaczana lokalnie przez l. Przyjęto kon-

wencję, że etykiety będą się składać z nazwy reguły reakcji z indeksem górnym, wskazującym na numer transformowanego przejścia.

(b) Wykorzystując pierwsze dwa elementy składowe przejścia (elementy zbiorów *Apl* i *Agt*) konstruowany jest łuk w multigrafie skierowanym reprezentującym przestrzeń stanów. Nowo skonstruowany łuk oznaczany będzie przez $e_x \quad x \in \mathbb{N}$.

(c) Dziedzina funkcji *res* i *par* przekształcanego przejścia zostaje ograniczona do identyfikatorów wierzchołków, których typy należą do zbioru $\mathbb{O}$.

(d) Wykorzystując składnik przekształcanego przejścia należący do zbioru *Apl* wyodrębniana jest etykieta czynności, reprezentowanej przez konstruowaną funkcję postępu misji. Etykieta ta będzie oznaczana przez *lab*.

(e) Zarówno dziedzina jak i przeciwdziedzina funkcji *res* zostają przekształcone tak, aby ich elementami były kolejne liczby naturalne dodatnie, zachowując przy tym relację porządku pomiędzy elementami dziedziny i przeciwdziedziny. Przykładowo funkcja $res = \{(0,4),(5,1),(2,17)\}$ zostanie przekształcona w $r\check{e}s = \{(1,2),(3,1),(2,3)\}$.

(f) Transformacja jakiej poddano elementy przeciwdziedziny funkcji *res* zostaje zaaplikowana do przeciwdziedziny funkcji *par*. Przeciwdziedzina funkcji *res* z poprzedniego kroku została przekształcona z użyciem funkcji $\phi = \{(4,2),(1,1),(17,3)\}$. Wiedząc to, przykładowa funkcja $par = \{(0,17),(2,4)\}$ zostanie przekształcona w $p\check{a}r = \{(0,3),(2,2)\}$.

(g) Forma funkcji postępu misji jest zależna od tego, ile obiektów bierze udział w czynności przez nią reprezentowanej. Rozróżniane są dwie sytuacje. Pierwsza, gdy w czynności bierze udział jeden obiekt, a druga, gdy uczestników jest więcej. W pierwszym przypadku funkcja postępu misji będzie miała postać:

$$f_x(c,t) = \begin{cases} c' & \text{jeżeli} \quad c = ((v_1,t_1),\ldots,(v_{n_o},t_{n_o}),\mathbb{H}') \\ 0 & \text{jeżeli} \quad c = 0 \end{cases}$$

W drugim przypadku (zaangażowanie więcej niż jednego obiektu) funkcja postępu misji będzie miała postać:

$$f_x(c,t) = \begin{cases} c' & \text{jeżeli} \quad c = ((v_1,t_1),\ldots,(v_{n_o},t_{n_o}),\mathbb{H}') \wedge \forall i \in \mathit{codom}(\mathit{păr})\ (v_i,z) \\ 0 & \text{w pozostałych przypadkach} \end{cases}$$

gdzie:

- n_o - liczba obiektów w zadaniu;
- $v_i, t_i, t, z \in \mathbb{N}$;
- $c' = \big((v'_1,t'_1),\ldots,(v'_{n_o},t'_{n_o}),\mathbb{H}' \cup \{l_{t+1}\}\big)$;
- $v'_i = v_{\mathit{rěs}(i)}$;
- $t'_i = t_{\mathit{rěs}(i)} + [\ \mathit{rěs}(i) \in \mathit{codom}(\mathit{păr})\ ] \cdot \mathit{time}(\mathit{lab})$;
- $\mathbb{H}' \in 2^{\mathbb{H}}$ gdzie $\mathbb{H}$ to zbiór zdefiniowany w 3.2.

Różnica pomiędzy powyższymi funkcjami postępu misji polega na tym, że w przypadku zaangażowania kilku obiektów, wszystkie z nich (przeciwdziedzina funkcji *păr*) muszą być w tym samym momencie czasu (jakaś wartość z).

(h) Nowo skonstruowana funkcja f_x przypisywana jest do łuku e_x tzn. $m_f \uplus \{(e_x, f_x)\}$.

Jako przykład rozpatrzony będzie proces przekształcenia dwóch przejść z TTS. Źródłowy TTS został przedstawiony w tabeli 2.2. Przestrzeń stanów powstała w wyniku przekształcenia wszystkich przejść z tej tabeli została zamieszczona w tabeli 3.1. Przedstawiony tutaj przykład zostanie ograniczony do pierwszego i trzeciego przejścia z 2.2 tj.:

Apl	Agt	Par	Res
$\langle B_0, r_1\rangle$	B_1	$\{(0,0),(1,1),(2,3)\}$	$\{(0,0),(1,2),(2,3),(3,1)\}$
$\langle B_0, r_2\rangle$	B_2	$\{(0,0),(1,1),(2,2),(3,3)\}$	$\{(0,0),(1,3),(2,1),(3,2)\}$

Zgodnie z procedurą przekształcania TTS w SS kolejne kroki tego procesu będą następujące:

1. W zadaniu występują jednocześnie dwa obiekty w postaci przemieszczających się dronów, stąd $n_o = 2$.

2. Definiowany jest zbiór typów wierzchołków, reprezentujących obiekty zadania $\mathbb{O} = \{U\}$.

3. Dwugrafy ze zbioru *Agt* przekształcane są w stany systemu. Wynikiem jest funkcja $states = \{(B_0, 0), (B_1, 1), (B_2, 2)\}$.

4. Określany jest czas potrzebny na wykonanie czynności reprezentowanych przez reguły reakcji: $time = \{(r_1, 2), (r_2, 1)\}$.

5. Przekształcane jest pierwsze przejście z TTS zdefiniowane w tabeli powyżej.

   (a) Etykieta identyfikująca przekształcane przejście ma postać $l = r1^1$. Wybór takiej etykiety jest podyktowany tym, że wykonywana jest czynność typu r_1, a przekształcane jest pierwsze przejście z TTS z tabeli 2.2.

   (b) Nowy łuk w konstruowanej przestrzeni stanów ma postać: $e_1 = (0, 1)$.

   (c) Funkcja $par = \{(0, 0), (1, 1), (2, 3)\}$ zostaje ograniczona do $par = \{(1, 1)\}$, a $res = \{(0, 0), (1, 2), (2, 3), (3, 1)\}$ do $res = \{(1, 2), (3, 1)\}$.

   (d) Etykieta czynności reprezentowanej przez przekształcane przejście to $lab = r_1$.

   (e) $r\check{e}s = \{(1, 2), (2, 1)\}$

   (f) $p\check{a}r = \{(1, 1)\}$

   (g) Czynność r_1 angażuje jeden obiekt zadania, ponieważ przeciwdziedzina funkcji $p\check{a}r$ ma jeden element. W związku z czym funkcja postępu misji ma postać:

   $$f_1(c, t) = \begin{cases} \left((v_2, t_2), (v_1, t_1 + 2), \mathbb{H}' \cup \{r1^1_{t+1}\}\right) & \text{jeżeli} \quad c = ((v_1, t_1), (v_2, t_2), \mathbb{H}') \\ 0 & \text{jeżeli} \quad c = 0 \end{cases}$$

   co jest równoważne funkcji f_1 z tabeli 3.1 (zmienne symboliczne a, b są w powyższej funkcji zapisane jako v_1, v_2, a x, y odpowiadają t_1, t_2).

   (h) $m_f = \{(e_1, f_1)\}$

6. Przekształcając kolejne przejście (czyli trzecie w tabeli 2.2) otrzymujemy:

   (a) $l = r2^3$,

   (b) $e_2 = (0, 2)$,

   (c) $par = \{(1, 1), (2, 2)\}$, $res = \{(2, 1), (3, 2)\}$,

   (d) $lab = r_2$,

   (e) $r\check{e}s = \{(1, 1), (2, 2)\}$,

   (f) $p\check{a}r = \{(1, 1), (2, 2)\}$ (bez zmian względem par) ,

   (g) Czynność r_2 angażuje więcej niż jeden obiekt ($\overline{codom(p\check{a}r)} = 2$) więc używany jest drugi szablon funkcji postępu misji:

$$f_3(c,t) = \begin{cases} \big((v_1, t_1 + 1), (v_2, t_2 + 1), \mathbb{H}' \cup \{r2^3_{t+1}\}\big) & \text{jeżeli} \quad c = ((v_1, t_1), (v_2, t_2), \mathbb{H}') \wedge ((v_1, z) \wedge (v_2, z \\ 0 & \text{w pozostałych przypadkach} \end{cases}$$

   co jest równoważne funkcji f_3 z tabeli 3.1 ;

   (h) $m_f = \{(e_1, f_1), (e_2, f_3)\}$.

Załącznik D

Konstrukcja stanu scenariusza

Tematem tego załącznika będzie sposób wyznaczenia przejścia w TTS, które posłuży do przekształcenia stanu systemu, w ramach procesu wyznaczenia stanu scenariusza w zadanym momencie czasu. Można też określić ten proces jako mający na celu znalezienie czynności (w postaci przejścia w TTS) wiedząc, jaki jest jej typ, stan początkowy, oraz które elementy zadania brały w niej udział.

Główną trudnością związaną z powyższym procesem jest to, że szukane przejście nie odpowiada temu, z którego skonstruowano funkcję postępu misji, będącą parametrem procesu konstrukcji stanu scenariusza.

Niżej opisana procedura odpowiada funkcji *Update* z etapu 4 algorytmu weryfikacji modelu MAS. Pomimo, że śledzący system przejść nie jest argumentem tej funkcji, to należy traktować przejścia (czyli elementy zbioru Tra) jako dostępne.

Parametrami wejściowymi tej funkcji są:

- $s \in Agt$ - dotychczas skonstruowany dwugraf reprezentujący stan systemu w momencie czasu, będącym parametrem etapu 3. W ramach tej iteracji etapu 4 jest on aktualizowany/rozbudowywany.

- $m_s \in M_{\mathcal{E}}$ - funkcja przypisująca unikalne identyfikatory elementów zadania do wierzchołków s.

- $r \in \mathcal{R}$ - reguła reakcji odpowiadająca czynności, która ma zostać wykonana w systemie.

- $m_r \in M_{\mathcal{E}}$ - funkcja przypisująca unikalne identyfikatory elementów zadania do wierzchołków redeksu r.

- n_x - pierwszy nowy unikalny identyfikator, który będzie przypisany nowemu elementowi otoczenia. Nowe elementy otoczenia mogą być wynikiem wykonania czynności reprezentowanej przez regułę reakcji r.

Funkcja *Update* działa według poniższego algorytmu:

1. Rozpatrywane będą tylko te przejścia w TTS, których element należący do zbioru *Apl* jest równy (s, r_x), gdzie $r_x \in Lab$ jest etykietą reguły reakcji r. Zbiór przejść spełniających te ograniczenia będzie oznaczany przez $Tra_{(s,r)}$.

2. Spośród elementów zbioru $Tra_{(s,r)}$ wybierany jest element, którego funkcja *par* spełnia zależność:

$$par \circ m_r \subset m_s$$

3. Nowy stan systemu (czyli s po wykonaniu czynności r, której uczestnicy są wskazywani przez złożenie funkcji $par \circ m_r$) jest reprezentowany przez element wybranego w poprzednim kroku przejścia, należący do zbioru Agt. W ramach tego rozdziału będziemy go oznaczać przez B_u.

4. Funkcja przypisująca unikalne identyfikatory elementów zadania do stanu systemu, reprezentowanego przez dwugraf wskazany w poprzednim kroku, jest konstruowana z dwóch składników:

   - Funkcji, która przypisuje unikalne identyfikatory elementów zadania istniejących przed wykonaniem czynności, do odpowiadających im wierzchołków w nowo skonstruowanym dwugrafie:

$$residue = \{(x, v) \mid \forall v \in dom(res) \quad m_s(x) = res(v)\}$$

   - Funkcji, która przypisuje unikalne identyfikatory, zaczynając od n_x, elementów zadania, które powstały w wyniku wykonania czynności, do odpowiadających im wierzchołków w nowo skonstruowanym dwugrafie B_u :

$$new = \{(x, v) \mid \forall v \in (V_{B_u} \setminus dom(res)) \quad (n_x + \overrightarrow{V_{B_u} \setminus dom(res)}\downarrow (v), v)\}$$

Obydwie funkcje stanowią składniki wynikowej funkcji res_m tzn.:

$$res_m = residue \uplus new$$

5. Nowa wartość n_x jest równa $n_x + \overline{V_{B_u} \setminus dom(res)}$.

Jako przykład użyty zostanie fragment przykładowej weryfikacji modelu z rozdziału 5.2. Rozpatrzone będą dwa wykonania funkcji *Update*. Pierwsze, będące w trakcie wykonywania po raz pierwszy etapu 4 w pierwszej iteracji etapu 3. Na podstawie tabeli 5.7 wiemy, że parametry wejściowe funkcji *Update* będą w tym momencie następujące:

- s - dwugraf reprezentujący stan systemu, który jest rozbudowywany w celu zdefiniowania stanu scenariusza w zadanym momencie czasu. W tym przypadku jest to stan początkowy systemu, oznaczany dalej jako $B0$. Został on przedstawiony na rysunku 5.5;
- $m_s = \{(0,0),(1,1),(2,2),(3,3),(4,4)\}$ - ponieważ mapowanie UI na wierzchołki stanu początkowego polega na przypisaniu kolejnych liczb naturalnych do kolejnych wierzchołków dwugrafu reprezentującego ten stan;
- r - reguła reakcji $r1$ (o takiej samej etykiecie);
- $m_r = \{(0,0),(2,1),(3,2)\}$;
- $n_x = 5$.

Przebieg funkcji *Update* wywołanej z użyciem powyższych argumentów będzie następujący:

1. Zbiór $Tra_{(B0,r1)}$ składa się z następujących elementów:

Apl	Agt	Par	Res
$\langle B0, r1\rangle$	$B1$	$\{(0,0),(1,2),(2,3)\}$	$\{(0,0),(1,3),(2,2),(3,1),(4,4)\}$
$\langle B0, r1\rangle$	$B1$	$\{(0,0),(1,1),(2,3)\}$	$\{(0,0),(1,3),(2,1),(3,2),(4,4)\}$

2. Ze zbioru przejść zdefiniowanego w poprzednim etapie wyszukiwane jest przejście, które spełnia zależność:

$$par \circ m_r \subset m_s$$

czyli:

$$par \circ \{(0,0),(2,1),(3,2)\} \subset \{(0,0),(1,1),(2,2),(3,3),(4,4)\}$$

Spośród dwóch dostępnych przejść, spełniającym powyższą zależność jest to z funkcją uczestnictwa równą:

$$par = \{(0,0),(1,2),(2,3)\}$$

Przejście to jest tym samym, na podstawie którego powstała funkcja postępu misji f_1, która z kolei jest elementem marszruty aktualnie wykorzystywanym do skonstruowania stanu scenariusza.

3. Nowy stan systemu ma postać dwugrafu $B1$.

4. Funkcja przypisująca unikalne identyfikatory do wierzchołków dwugrafu $B1$ ma postać:
$$res_m = \{(0,0),(3,1),(2,2),(1,3),(4,4)\} \uplus \emptyset$$

5. $n_x = 5$ - ponieważ w wyniku czynności $r1$ nie powstał żaden nowy element otoczenia.

Drugi przykład będzie przedstawiał przebieg wykonywania funkcji *Update* w trakcie wykonania etapu 4, w ramach trzeciej iteracji etapu 3 konstruującego stan scenariusza w momencie czasu równym 1. Jest to następne wykonanie funkcji *Update* po tym, które zostało omówione wcześniej w tym rozdziale. W związku z tym, argumenty wejściowe dla tego wywołania omawianej funkcji są następujące:

- s - dwugraf będący stanem systemu, który jest rozbudowywany w celu zdefiniowania stanu scenariusza w zadanym momencie czasu. W tym przypadku jest to dwugraf, będący wynikiem poprzedniego wykonania funkcji *Update*. Oznaczać go będziemy jako $B1$. Postać graficzna tego dwugrafu została rzedstawiona na rysunku 5.6;

- $m_s = \{(0,0),(3,1),(2,2),(1,3),(4,4)\}$ - mapowanie UI na wierzchołki stanu systemu, będącego wynikiem poprzedniego wykonania funkcji *Update*;

- r - reguła reakcji $r1$ (o takiej samej etykiecie);
- $m_r = \{(0,0),(1,1),(3,2)\}$;
- $n_x = 5$.

Przebieg funkcji *Update* wywołanej z użyciem powyższych argumentów będzie następujący:

1. Zbiór $Tra_{(B1,r1)}$ składa się z następujących elementów:

Apl	Agt	Par	Res
$\langle B1, r1\rangle$	$B3$	$\{(0,0),(1,3),(2,1)\}$	$\{(0,0),(1,1),(2,2),(3,3),(4,4)\}$

2. Ze zbioru przejść zdefiniowanego w poprzednim etapie wyszukiwane jest przejście, które spełnia zależność:

$$par \circ m_r \subset m_s$$

czyli:

$$par \circ \{(0,0),(1,1),(3,2)\} \subset \{(0,0),(3,1),(2,2),(1,3),(4,4)\}$$

Z uwagi na to, że „wybieramy" spośród tylko jednego przejścia, oczywistym jest, które przejście zostanie użyte. Warto tutaj zaznaczyć, że jest to inne przejście niż to, którego użyto do konstrukcji funkcji postępu misji f_5. Innymi słowy, funkcja f_5, z użyciem której konstruowany jest teraz stan scenariusza, została opracowana na podstawie piątego przejścia w TTS z tabeli 5.5, podczas gdy przejście czwarte w tym samym TTS zawiera informacje o zmianach, jakim trzeba poddać dwugraf $B1$.

3. $res_m = \{(0,0),(3,1),(2,2),(1,3),(4,4)\}$.
4. $n_x = 5$.

Załącznik E

Analiza złożoności wybranych algorytmów

Ten załącznik poświęcony będzie wyznaczeniu złożoności obliczeniowej operacji, związanych z konstrukcją wszystkich marszrut o zadanej długości, w przestrzeni stanów.

Przed przystąpieniem do analizy wspomnianych wyżej operacji wprowadzone będą następujące pojęcia:

- *Pesymistyczny czas działania algorytmu* zależny od rozmiaru danych wejściowych n oznaczany jako $t(n)$. Jest to funkcja, której argumenty należą do zbioru liczb naturalnych. Definiuje ona największą możliwą liczbę operacji elementarnych, wykonywanych w trakcie działania algorytmu, któremu podano na wejście dowolne dane rozmiaru n.

- *Pesymistyczna złożoność pamięciowa algorytmu* zależna od rozmiaru danych wejściowych n oznaczana jako $m(n)$. W tym rozdziale będzie to największy możliwy rozmiar wyniku działania algorytmu, dla dowolnych danych wejściowych rozmiaru n.

- *Asymptotyczne ograniczenie górne* oznaczane jako O oznacza zbiór funkcji, których wszystkie wartości dla dostatecznie dużych argumentów wejściowych będą nie mniejsze od wartości funkcji parametryzującej ten zbiór.

Formalnie zbiór O parametryzowany funkcją $g(n)$ zdefiniowany będzie jako:

$$O(g(n)) = \{f(n) \mid \exists c, n_0 \forall n > n_0 \quad 0 \leq f(n) \leq c \cdot g(n)\}$$

Zapis $t(n) \in O(g(n))$ należy rozumieć jako *pesymistyczny czas działania algorytmu jest ograniczony od góry przez funkcję* $g(n)$. Można to interpretować jako stwierdzenie, że maksymalna liczba operacji wykonanych w trakcie działania omawianego algorytmu, dla dostatecznie dużych danych, nie przekroczy wartości, jaką przyjmie funkcja g (wymnożona przez jakiś współczynnik c).

Procedurą, której złożoność będzie rozpatrywana w tym rozdziale jest algorytm wyznaczania wszystkich marszrut zadanej długości w przestrzeni stanów, omówiony w rozdziale 3.3. Analizę tego algorytmu rozpoczniemy od procesu wyznaczania wszystkich marszrut o jeden krok dłuższych niż aktualnie znane. Proces ten można zdekomponować na dwie procedury. Pierwszą jest wyznaczanie splotu szeregów zdefiniowanych we wspomnianym rozdziale. Algorytm tej procedury można przedstawić następująco:

```
1  function CONVOLUTE (mk,mf,s,t) =
2    result=[]
3    n_s=mk.cols // atrybut cols zwraca liczbę kolumn
4    for i=0 to n_s - 1 do
5      walks_to_i = mk.get(i) // przebiegi misji do stanu „i''
6      f_i_s = mf.get(i,s) // funkcje przejścia z „i'' do „s''
7      if walks_to_i.length == 0 || f_i_s == f_null then continue
8      // continue kończy aktualną iterację, ale nie przerywa pętli
9      for j=0 to f_i_s.length - 1 do
10       f=f_i_s.get(j) // j-ta funkcja postępu misji
11       for k=0 to walks_to_i.length - 1 do
12         w=walks_to_i.get(k) // k-ty przebieg misji do i-tego stanu
13         if f(w,t+1) != 0 then
14           f(w,t+1)::result
15   return result
```

Algorytm E.1: Algorytm wyznaczania splotu szeregów K_i^t z szeregami $F_{i,s}$.

Funkcja `CONVOLUTE` przyjmuje cztery argumenty: macierz `mk` aktualnie znanych

przebiegów misji ($\mathbb{M}_K^t$ według oznaczeń z rozdziału 3.3), macierz `mf` wszystkich funkcji postępu misji (oznaczana jako $\mathbb{M}_F^t$ w definicjach formalnych), stan systemu do którego poszukiwane są nowe przebiegi misji oraz długość marszrut definiujących te przebiegi. Sposób działania tej funkcji można opisać następująco. Dla kolejnych stanów systemu (**for** w linii 4) sprawdzane jest, czy istnieją jakiekolwiek przebiegi misji doprowadzające do tego stanu oraz czy istnieją funkcje postępu misji do stanu `s` (**if** w linii 7). Jeżeli tak, to wartość kolejnych funkcji postępu misji niebędąca elementem neutralnym (warunek sprawdzany w linii 13) dla wszystkich tych przebiegów misji (**for** w linii 11) oraz argumentu `t` jest dodawana do zmiennej `result` (linia 14). Zmienna `result` będąca rezultatem funkcji `CONVOLUTE` może być typu reprezentującego listy jednokierunkowe. Jest on dostępny we wszystkich obecnie dominujących funkcyjnych językach programowania. Użycie tego typu do reprezentacji wyniku jest podyktowane tym, że jego rozbudowa (operacja `::`) ma stałą złożoność czasową.

Drugą operacją, która składa się na proces wyznaczania marszrut o jeden krok dłuższych od aktualnie znanych jest proces mnożenia macierzy $\mathbb{M}_K^t \cdot \mathbb{M}_F^t$. Wykorzystuje on funkcję `CONVOLUTE` zdefiniowaną powyżej w celu wyznaczenia kolejnych elementów macierzy wynikowej. Definicja algorytmu mnożenia macierzy $\mathbb{M}_K^t \cdot \mathbb{M}_F^t$ jest następująca:

```
1  function MULTIPLY_MATRICES_OF_SEQ (mk,mf,t) =
2    cols=mk.cols
3    result = new Matrix(1,cols)
4    // wynikiem jest macierz o jednym wierszu i cols kolumnach
5    for s=0 to cols - 1 do
6      partial_result=CONVOLUTE(mk,mf,s,t)
7      result.insertAt(i,partial_result)
8    return result
```

Algorytm E.2: Algorytm mnożenia macierzy $\mathbb{M}_K^t$ przez macierz $\mathbb{M}_F^t$.

Przechodząc do analizy złożoności obliczeniowej powyższych operacji należy zacząć od zdefiniowania, które operacje będziemy zliczać. Dla funkcji `CONVOLUTE` będzie to sprawdzenie dwóch warunków w linii 7, operacja sprawdzenia czy wynik funkcji postępu misji z linii 13 jest elementem neutralnym oraz rozszerzenie wyniku wykonywane w linii 14. Pesymistyczną złożoność czasową dla danych

wejściowych `mk,mf,s,t` można przedstawić jako następującą sumę:

$$(2 + 2 \cdot c_1 \cdot f_{1,s}) + \quad \cdots \quad + (2 + 2 \cdot c_{n_s} \cdot f_{n_s,s}) = \sum_{i=1}^{n_s} (2 + 2 \cdot c_i \cdot f_{i,s})$$

Bez względu na postać macierzy $\mathbb{M}_F^t$ i $\mathbb{M}_K^t$ dwie instrukcje warunkowe w linii 7 będą zawsze sprawdzane, stąd w każdym składniku sumy znajduje się dwójka. W zależności od liczby przebiegów misji do każdego stanu (c_i) użytych jako argument funkcji postępu misji z tego stanu do stanu s ($f_{i,s}$), tyle razy będzie wykonana instrukcja warunkowa z linii 13 oraz rozszerzenie wyniku w linii 14. Wartość n_s oznacza liczbę wszystkich stanów w systemie. Powyższa suma jest zależna od trzech parametrów:

- Sumy wszystkich przebiegów misji (do dowolnego stanu systemu) - n_c.
- Sumy funkcji postępu misji z dowolnego stanu do stanu s - n_{f_s}.
- Liczby stanów w systemie - n_s.

Z kolei wiedząc, że funkcja `MULTIPLY_MATRICES_OF_SEQ` sprowadza się do wywołania `CONVOLUTE` n_s razy (dla każdego stanu systemu) i zapisania wyniku w odpowiedniej kolumnie macierzy $\mathbb{M}_K^{t+1}$, można zdefiniować jej pesymistyczną złożoność czasową jako:

$$\sum_{s=1}^{n_s} 2 \cdot \sum_{i=1}^{n_s} (2 + 2 \cdot c_i \cdot f_{i,s}) = 4n_s^2 + 4 \cdot \sum_{s=1}^{n_s} \sum_{i=1}^{n_s} (c_i \cdot f_{i,s})$$

Sumę wszystkich funkcji postępu misji można zapisać jako: $\sum_{s=1}^{n_s} \sum_{i=1}^{n_s} f_{i,s} = n_f$, a suma przebiegów misji dla wszystkich stanów będzie określona zależnością[1] $\sum_{s=1}^{n_s} n_K(K_s^t) = n_c$.

Powyżej zdefiniowana pesymistyczna złożoność obliczeniowa procedury mnożenia macierzy $\mathbb{M}_K^t \cdot \mathbb{M}_F^t$ jest zależna od trzech parametrów: liczby wszystkich stanów systemu n_s, sumy wszystkich funkcji postępu misji n_f oraz sumy wszystkich aktualnie znanych przebiegów misji o długości t, którą oznaczamy przez n_c. Złożoność ta będzie dalej oznaczana jako $t(n_s, n_f, n_c)$.

[1]Funkcja n_K została zdefiniowana w rozdziale 3.3 i zwraca liczbę elementów szeregu podanego jako argument.

Wartość funkcji $t(n_s, n_f, n_c)$ będzie największa w przypadku, gdy wszystkie aktualnie znane przebiegi misji będą prowadziły do jednego stanu, z którego wszystkie funkcje postępu misji umożliwią przejście do dowolnego innego stanu systemu. Innymi słowy, w celu wyznaczenia macierzy $\mathbb{M}_K^{t+1}$ trzeba będzie wykonać najwięcej operacji, gdy macierz $\mathbb{M}_K^t$ będzie zawierała tylko jeden szereg (kolumnę) nie będący elementem neutralnym, a macierz $\mathbb{M}_F^t$ tylko jeden wiersz, którego elementami nie będą szeregi składające się z funkcji f_{null}. Asymptotyczne ograniczenie górne pesymistycznego czasu działania algorytmu wyznaczania macierzy $\mathbb{M}_K^{t+1}$ można więc zdefiniować jako:

$$t(n_s, n_f, n_c) \in O((n_s)^2 + n_c \cdot n_f)$$

Przechodząc do pesymistycznej złożoności pamięciowej, chcemy określić jaka jest największa możliwa suma przebiegów misji, różnych od 0, będących wynikiem pojedynczego mnożenia macierzy $\mathbb{M}_K^t \cdot \mathbb{M}_F^t$. Korzystając z pesymistycznej złożoności czasowej zdefiniowanej powyżej wiemy, że najwięcej nowych przebiegów misji będzie wyliczanych w przypadku, gdy wszystkie aktualnie znane przebiegi będą w jednej kolumnie macierzy $\mathbb{M}_K^t$, a wszystkie funkcje postępu w jednym wierszu $\mathbb{M}_B^t$. W takiej sytuacji największa[2] liczba nowych przebiegów misji w wyniku iloczynu wspomnianych macierzy będzie równa $n_c \cdot n_f$. W związku z tym asymptotyczne ograniczenie górne złożoności pamięciowej można zdefiniować jako:

$$m(n_f, n_c) \in O(n_f \cdot n_c)$$

Obydwa asymptotyczne oszacowania pozwalają na przystąpienie do analizy złożoności czasowej procesu wyznaczania wszystkich polityk zachowania, składających się z n czynności. Tak jak zostało to opisane w rozdziale 3.3 proces wyznaczenia polityki zachowania o długości n polega na wykonaniu iloczynu $\mathbb{M}_K^{t+1} = \mathbb{M}_K^t \cdot \mathbb{M}_F^t$ n razy, z czego przy każdym kolejnym mnożeniu wartość macierzy $\mathbb{M}_K^t$ jest zastępowana przez wynik poprzedniego. Dla l takich iloczynów pesymistyczną złożoność czasową takiej operacji można zapisać jako zależność

[2]W sytuacji, gdy część funkcji postępu misji reprezentuje czynności kooperacyjne, liczba przebiegów misji może być mniejsza, co można zaobserwować analizując przykład z rozdziału 5.1.

rekurencyjną:

$$t^{(l)}(n_s, n_f, n_c) \in \begin{cases} O((n_s)^2 + n_c \cdot n_f) & \text{jeżeli } n = 1 \\ t^{(l-1)}(n_s, n_f, n_f \cdot n_c) + O((n_s)^2 + n_c \cdot n_f) & \text{jeżeli } n > 1 \end{cases}$$

Powyższa zależność uwzględnia fakt, że w pesymistycznym przypadku po każdej iteracji mnożenia, wynikowa macierz będzie zawierać $n_f \cdot n_c$ przebiegów misji. Dla przykładu rozpatrzmy powyższą zależność dla $l = 3$. Będzie ona definiować asymptotyczne ograniczenie górne złożoności czasowej operacji wyznaczenia polityki zachowania o długości 3, w przestrzeni stanów składającej się z n_s stanów, n_f liczby funkcji postępu misji oraz n_c początkowych (w macierzy $\mathbb{M}_K^0$) przebiegów misji. Powyższa zależność rekurencyjna dla wymienionych parametrów przedstawiona jako suma, będzie miała postać:

$$O((n_s)^2 + n_c \cdot n_f) + O((n_s)^2 + (n_c \cdot n_f) \cdot n_f) + O((n_s)^2 + (((n_c \cdot n_f)) \cdot n_f) \cdot n_f)$$

W ogólnym przypadku dla polityki zachowania o długości l powyższa zależność rekurencyjna przyjmie postać sumy:

$$O(n_s^2 + n_c \cdot n_f) + O(n_s^2 + (n_c \cdot (n_f)^2)) + \cdots + O((n_s)^2 + (n_c \cdot (n_f)^l)) = \sum_{i=1}^{l} (O((n_s)^2 + n_c \cdot (n_f)^l))$$

Ostatnią sumę w powyższej zależności można przedstawić jako:

$$O(\sum_{i=1}^{l}((n_s)^2 + n_c \cdot (n_f)^l)) = O(l \cdot n_s^2 + \sum_{i=1}^{l}(n_c \cdot (n_f)^l)) = O(l \cdot (n_s)^2 + n_c \cdot \frac{(n_f)^{l+1} - 1}{n_f - 1})$$

Znając powyższe można ograniczyć pesymistyczny czas działania algorytmu wyznaczania polityk zachowania (czyli marszrut) w przestrzeni stanów, składającej się z n_s stanów, gdzie suma funkcji postępu misji wynosi n_f, a liczba początkowych przebiegów misji jest równa n_c:

$$t^{(l)}(n_s, n_f, n_c) \in O(l \cdot (n_s)^2 + n_c \cdot \frac{(n_f)^{l+1} - 1}{n_f - 1})$$

Załącznik F

Rozszerzanie marszruty

Celem tego złącznika jest zaprezentowanie, w jaki sposób przebiega proces rozszerzania marszruty w przestrzeni stanów. Jest on wykonywany w ramach etapu 2 metody weryfikacji modeli MAS. Schemat blokowy omawianego tutaj algorytmu został przedstawiony w rozdziale 4.2.3.

Operacja wykonywana w ramach etapu 2 nie może zakończyć się wykryciem błędu, przez co jej przebieg był pomijany we wcześniejszych przykładach. Spośród wszystkich kroków wymienionych na schemacie 4.4 wyjaśnienia mogą wymagać cztery, są nimi:

- $u = Corr_{Rel}(f_t)$ - funkcja $Corr_{Rel} : T \to Rel$ przekształca funkcję postępu misji (element zbioru T z SS) we względną zmianę konfiguracji SaT. Funkcje o podobnym celu są wyznaczane w ramach przekształcania przejść z TTS w funkcje postępu misji i zostały opisane w załączniku C.
  Dla przykładu funkcja postępu misji:

$$f_t(c,t) = \begin{cases} [\langle (b,y),(a,x+2)\rangle, \mathbb{H}' \cup \{rx_{t+1}\}] & c = [\langle (a,x),(b,y)\rangle, \mathbb{H}'] \\ 0 & c = 0 \end{cases}$$

  Zostanie przekształcona w funkcję:

$$Corr_{Rel}(f_t) = u = ((2,0),(1,2))$$

- $i_n = Time_{Rel}(i_c, u)$ - funkcja $Time_{Rel} : I \times Rel \to I$ przekształca konfigura-

cję SaT, w oparciu o przekształcenie względne wyznaczone w poprzednim punkcie.

Przykładowo dla SaT:

$$i = \langle (4,7), (1,2), (2,7), (3,1) \rangle$$

i przekształcenie względne $u = ((2,0), (1,1), (4,0), (3,1))$ Wynikiem funkcji $Time_{Rel}(i,u)$ będzie:

$$\langle (1,2), (4,8), (3,1), (2,8) \rangle$$

- $A_d, n_d = Time_D(i_c, i_n)$ - funkcja ta wyznacza identyfikatory obiektów z konfiguracji SaT, dla których czas uległ zmianie oraz wartość, o jaką nastąpiła ta zmiana.

  Przykładowo dla pary konfiguracji SaT z poprzedniego punktu tj.:

$$i_c = \langle (4,7), (1,2), (2,7), (3,1) \rangle$$

$$i_n = \langle (1,2), (4,8), (3,1), (2,8) \rangle$$

  Funkcja $Time_D(i_c, i_n)$ zwróci jako wynik:

$$A_d = \{4,2\} \quad n_d = 1$$

- $m_r, m'_{full}, n'_x = Trans(t_{tra}, m_{full}, n_x)$ - funkcja przypisująca unikalne identyfikatory zarówno do wierzchołków redeksu, jak i do dwugrafu, będącego stanem wyjściowym przejścia TTS podanego na wejściu. Ponadto funkcja ta zwraca pierwszy (najmniejszy) nieużyty unikalny identyfikator.

  Funkcja m_r będąca wynikiem $Trans(t_{tra}, m_{full}, n_x)$ jest definiowana jako:

$$m_r = \{(x, par^{-1} \circ m_{full}(x) \mid \forall x \in dom(m_{full}) \quad par \in t_{tra}\} \uplus new_r$$

Z kolei m'_{full} będąca wynikiem tej samej funkcji jest definiowana jako:

$$m'_{full} = \{(x, res^{-1} \circ m_{full}(x)) \mid \forall x \in dom(m_{full}) \quad res \in t_{tra}\} \uplus new_{full}$$

W obydwu przypadkach funkcja *new* przypisuje „nowe" unikalne identyfikatory wierzchołkom, do których wcześniej nie przypisano unikalnego identyfikatora, w następujący sposób:

$$new_r = \{(v + n_x, v) \mid \forall v \in codom(par) \setminus codom(m_{full})\}$$

$$new_{full} = \{(v + n_x, v) \mid \forall v \in codom(res) \setminus codom(m_{full})\}$$

Wartość n'_x jest równa największej sumie $v+n_x$ zarówno w new_r lub new_{full}. W przypadku braku nowych elementów zadania n'_x jest równe n_x.

Mając zdefiniowane powyższe funkcje, można przystąpić do przedstawienia przebiegu etapu 2 weryfikacji poprawności modelu MAS, omówionego w rozdziale 5.2. Marszruta stanowiąca podstawę harmonogramu działań dla modelowanego systemu ma postać: $0 \xrightarrow{f_1} 1 \xrightarrow{f_3} 2 \xrightarrow{f_5} 4 \xrightarrow{f_8} 5$. Zbiór W odpowiadający tej marszrucie ma postać:

$$W = \{(1, f_1), (2, f_3), (3, f_5), (4, f_8)\}$$

Jako przypomnienie, w tabeli F.1 zamieszczono definicje funkcji postępu misji ze zbioru W. Wiedząc, że stan początkowy systemu w postaci dwugrafu odpowiada temu z rysunku 5.5, wartości początkowe zmiennych pomocniczych etapu 2 dla tego modelu są następujące:

- $n_x = 5$,
- $m_{full} = \{(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)\}$,
- $i_c = \langle (1, 0), (2, 0) \rangle$.

Przebieg etapu 2 algorytmu weryfikacji modelu MAS opisanego w skrócie powyżej, a szczegółowo w rozdziale 5.2, którego parametrami wejściowymi jest zbiór W został przedstawiony w tabeli F.2.

Tabela F.1: Funkcje postępu misji potrzebne do analizy przebiegu etapu 2. Definicje tutaj przedstawione są podzbiorem funkcji zamieszczonych w tabeli 5.6.

$f_1(c,t) \rightarrow$	$\langle (b, y+1), (a, x) \rangle \pm r1^1_{t+1}$
$f_3(c,t) \rightarrow$	$\langle (a, x+1), (b, y) \rangle \pm r2^3_{t+1}$
$f_5(c,t) \rightarrow$	$\langle (b, y+1), (a, x) \rangle \pm r1^5_{t+1}$
$f_8(c,t) \rightarrow$	$\langle (b, y), (a, x+1) \rangle \pm r2^8_{t+1}$

Tabela F.2: Przebieg etapu 2 algorytmu weryfikacji poprawności modelu MAS dla przykładu z rozdziału 5.2.

Etap	**Krok**	**Wynik/Komentarz**
2	$n_f, f_t, W_c = First(W_c)$	$n_f = 1$ $f_t = f_1$ $W_c = \{(2, f_3), (3, f_5), (4, f_8)\}$
2	$u = Corr_{Rel}(f_t)$	$u = ((2,1), (1,0))$
2	$i_n = Time_{Rel}(i_c, f_u)$	$i_n = \langle (2,1), (1,0) \rangle$
2	$A_d, n_d = Time_D(i_c, i_n)$	$A_d = \{2\}, n_d = 1$
2	$t_{tra} = Corr_{Tra}(f_t)$	t_{tra} = pierwszy wiersz tabeli 5.5
2	$m_r, m'_{full}, n'_x = Trans(t_{tra}, m_{full}, n_x)$	$par^{-1} = \{(0,0), (2,1), (3,2)\}$ $res^{-1} = \{(0,0), (3,1), (2,2), (1,3), (4,4)\}$ $m_r = \{(0,0), (2,1), (3,2)\}$ $m'_{full} = \{(0,0), (1,3), (2,2), (3,1), (4,4)\}$ $n'_x = 5$
2	$W_r = W_r \cup \{(n_f, f_t, m_r, m_{full}, n_x, (A_d, n_d))\}$	Należy zwrócić uwagę, że dodawany jest element m_{full}, a nie m'_{full}
2	$n_x = n'_x$	$n_x = 5$
2	$m_{full} = m'_{full}$	$m_{full} = \{(0,0), (1,3), (2,2), (3,1), (4,4)\}$
2	$n_f, f_t, W_c = First(W_c)$	$n_f = 2$ $f_t = f_3$ $W_c = \{(3, f_5), (4, f_8)\}$
2	$u = Corr_{Rel}(f_t)$	$u = ((1,1), (2,0))$
2	$i_n = Time_{Rel}(i_c, f_u)$	$i_n = \langle (2,2), (1,0) \rangle$
2	$A_d, n_d = Time_D(i_c, i_n)$	$A_d = \{2\}, n_d = 1$
Kontynuacja na następnej stronie		

Tabela F.2 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
2	$t_{tra} = Corr_{Tra}(f_t)$	t_{tra} = trzeci wiersz tabeli 5.5
2	$m_r, m'_{full}, n'_x = Trans(t_{tra}, m_{full}, n_x)$	$par^{-1} = \{(1,0),(2,1),(4,2)\}$ $res^{-1} = \{(1,0),(4,1),(2,2),(0,3),(3,4)\}$ $m_r = \{(2,1),(3,0),(4,2)\}$ $m'_{full} = \{(0,3),(1,4),(2,2),(3,0),(4,1)\}$ $n'_x = 5$
2	$W_r = W_r \cup \{(n_f, f_t, m_r, m_{full}, n_x, (A_d, n_d))\}$	
2	$n_x = n'_x$	$n_x = 5$
2	$m_{full} = m'_{full}$	$m_{full} = \{(0,3),(1,4),(2,2),(3,0),(4,1)\}$
2	$n_f, f_t, W_c = First(W_c)$	$n_f = 3$ $f_t = f_5$ $W_c = \{(4, f_8)\}$
2	$u = Corr_{Rel}(f_t)$	$u = ((2,1),(1,0))$
2	$i_n = Time_{Rel}(i_c, f_u)$	$i_n = \langle(1,1),(2,2)\rangle$
2	$A_d, n_d = Time_D(i_c, i_n)$	$A_d = \{1\}, n_d = 1$
2	$t_{tra} = Corr_{Tra}(f_t)$	t_{tra} = piąty wiersz tabeli 5.5
2	$m_r, m'_{full}, n'_x = Trans(t_{tra}, m_{full}, n_x)$	$par^{-1} = \{(3,0),(4,1),(0,2)\}$ $res^{-1} = \{(3,0),(0,1),(4,2),(1,3),(2,4)\}$ $m_r = \{(0,0),(1,1),(3,2)\}$ $m'_{full} = \{(0,0),(1,2),(2,4),(3,1),(4,3)\}$ $n'_x = 5$
2	$W_r = W_r \cup \{(n_f, f_t, m_r, m_{full}, n_x, (A_d, n_d))\}$	
2	$n_x = n'_x$	$n_x = 5$
2	$m_{full} = m'_{full}$	$m_{full} = \{(0,0),(1,2),(2,4),(3,1),(4,3)\}$
2	$n_f, f_t, W_c = First(W_c)$	$n_f = 4$ $f_t = f_8$ $W_c = \emptyset$
2	$u = Corr_{Rel}(f_t)$	$u = ((2,0),(1,1))$
Kontynuacja na następnej stronie		

Tabela F.2 – kontynuacja z poprzedniej strony

Etap	**Krok**	**Wynik/Komentarz**
2	$i_n = Time_{Rel}(i_c, f_u)$	$i_n = \langle(2,2),(1,2)\rangle$
2	$A_d, n_d = Time_D(i_c, i_n)$	$A_d = \{1\}, n_d = 1$
2	$t_{tra} = Corr_{Tra}(f_t)$	t_{tra} = ósmy wiersz tabeli 5.5
2	$m_r, m'_{full}, n'_x = Trans(t_{tra}, m_{full}, n_x)$	$par^{-1} = \{(1,0),(2,1),(3,2)\}$ $res^{-1} = \{(1,0),(3,1),(4,2),(0,3),(2,4)\}$ $m_r = \{(3,0),(1,1),(4,2)\}$ $m'_{full} = \{(0,3),(1,4),(2,2),(3,3),(4,1)\}$ $n'_x = 5$
2	$W_r = W_r \cup \{(n_f, f_t, m_r, m_{full}, n_x, (A_d, n_d))\}$	
2	$n_x = n'_x$	$n_x = 5$
2	$m_{full} = m'_{full}$	$m_{full} = \{(0,3),(1,4),(2,2),(3,3),(4,1)\}$
2	$W_c = \emptyset$	Koniec

Wynikiem etapu 2, którego przebieg zamieszczono w tabeli F.2 jest rozszerzona marszruta postaci $W_M = \{e_1, e_2, e_3, e_4\}$. Kolejne jej elementy mają następującą postać:

1. $e_1 = (1, f_1, \{(0,0),(2,1),(3,2)\}, \{(0,0),(1,1),(2,2),(3,3),(4,4)\}, 5, (\{2\}, 1))$,
2. $e_2 = (2, f_3, \{(2,1),(3,0),(4,2)\}, \{(0,0),(1,3),(2,2),(3,1),(4,4)\}, 5, (\{2\}, 1))$,
3. $e_3 = (3, f_5, \{(0,0),(1,1),(3,2)\}, \{(0,3),(1,4),(2,2),(3,0),(4,1)\}, 5, (\{1\}, 1))$,
4. $e_4 = (4, f_8, \{(3,0),(1,1),(4,2)\}, \{(0,0),(1,2),(2,4),(3,1),(4,3)\}, 5, (\{1\}, 1))$.